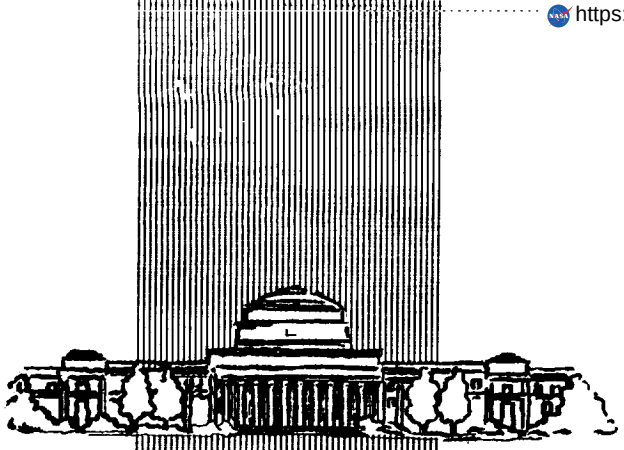


MASSACHUSETTS INSTITUTE OF TECHNOLOGY

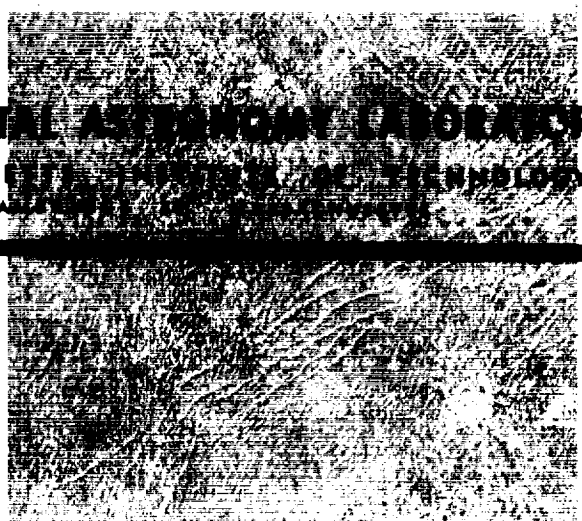
RE-19

A SPECIAL-PURPOSE INTERPLANETARY TRAJECTORY COMPUTATION PROGRAM FOR GUIDANCE AND NAVIGATION STUDIES

by

William T. McDonald

July 1965



EXPERIMENTAL ASTRONOMY LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MASSACHUSETTS

RE-19

A SPECIAL-PURPOSE INTERPLANETARY TRAJECTORY COMPUTATION PROGRAM FOR GUIDANCE AND NAVIGATION STUDIES

by

William T. McDonald

Experimental Astronomy Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts, 02139

Approved: W. Markey
Director,
Experimental Astronomy Laboratory

ACKNOWLEDGMENT

The program was prepared with the guidance and assistance of a number of persons at the M. I. T. Experimental Astronomy Laboratory. Professor James E. Potter guided the preparation of the program at all key stages, devised the new formulation of Kepler's problem for hyperbolic orbits, and directed the use of the Nystrøm integration technique and the preparation of the time step computation technique. Dr. Robert G. Stern and Mr. Gary L. Slater devoted much time and interest to fruitful discussions and contributed many helpful suggestions. Mr. William E. Margolis gave much programming advice and assistance.

Acknowledgment is made for the use of the IBM 7094 computer at the M. I. T. Computation Center. The development of this program was done as problem # M3938.

This effort was supported entirely by DSR Project 6145 through NASA Grant NsG 254-62.

RE-19

A SPECIAL-PURPOSE INTERPLANETARY TRAJECTORY COMPUTATION PROGRAM FOR GUIDANCE AND NAVIGATION STUDIES

ABSTRACT 30315

This report contains a technical description of an interplanetary trajectory computation program written specifically to be used in making comparative evaluations of interplanetary guidance and navigation techniques. The special requirements for this purpose are defined, and the corresponding capabilities of the program are explained. Chief among these are simultaneous state and state transition matrix integration; a fixed time of arrival, fixed target point trajectory search capability; computation of trajectories in either time direction; ease of modification by non-professional programmers; and ability to be batch-processed in a high speed, high volume computation center.

Some computational techniques developed in the preparation of this program which may have useful applications in other programs for other purposes are explained in detail. These include a linear trajectory search method based on the linearized state transition matrix, a new formulation of the hyperbolic Kepler problem for high precision conic computations, and a unique accuracy self-check method based on forward and backward trajectory integration.

Computational error sources are discussed and typical accuracy data are given. The program subroutine structure and functional flow are described briefly. A companion document, the program User's Manual, contains more detailed information about the program structure.

Data are presented to illustrate operation of the single-precision version of the program. A double-precision version is also available. Both single- and double-precision versions are available in FORTRAN II and FORTRAN IV languages.

William T. McDonald
July 1965

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

CHAPTER I

INTRODUCTION

This program has been written to implement studies of guidance and navigation techniques for interplanetary flight. It computes point mass interplanetary trajectories in a many-body gravitational field and other necessary data. The program serves as an evaluation standard in these studies: that is, it is first used to search for and establish nominal trajectories and to compute state transition matrices at points along these trajectories. Then it is used to compute, by means of the non-linear many body equations of motion, the target miss due to deviations measured or corrections made along a nominal trajectory according to the methods of the navigation or guidance techniques under study. The results evaluate the accuracies of the techniques.

The program requirements for these technique studies are fundamentally different from the requirements of other programs intended for spacecraft targeting or launch window analysis, and these different requirements have justified the cost of preparing this new program. Firstly, the program is required to compute the state transition matrix, in addition to the state vector, which is a fundamental matrix used in nearly all linearized guidance and navigation techniques. Secondly, since the evaluation of a technique requires multiple computations of effects of deviations or corrections for several different types of missions, the program must operate economically. This in turn requires not only high processing speed, but also that the program meet the requirements of high-production automatic batch processing at the MIT Computation Center. Thirdly, to allow for changing requirements of a variety of anticipated studies by engineering personnel, the program is required to have the capability of easy modification by non-professional programmers.

A number of tradeoffs have been necessary in designing the program to accomplish these requirements, since the requirements conflict to some extent. The program is written in FORTRAN II for the IBM 7094 computer at the MIT Computation Center, and a FORTRAN IV version has been prepared for the IBM System 360. The use of a more efficient

language, such as FAP, would enhance the processing speed considerably; but non-professional programmers would not be able to easily modify the program. The requirements of high-production batch processing are that the program operate with a standard monitor system and that no input data tapes (e.g. tape-stored ephemerides) be used. Programs which do not meet these requirements must be processed with special handling procedures, and this seriously curtails the use of such a program for studies involving a large amount of computation. However, the intended use of this program for guidance and navigation technique evaluations does not require highly precise planet ephemerides, and thus internal computation of ephemerides is permitted. The basic reason for this is that, as long as the solar system model used in the program is a reasonably realistic representation of the actual solar system, the absolute accuracy of a technique can be determined to within at least an order of magnitude and, furthermore, the relative accuracies of different techniques can be determined precisely. Accordingly, the present program uses a simplified solar system model in which the motion of the Sun is neglected, the included planets are Venus, Earth, Mars, Jupiter, and Saturn, the influences of the Moon and other solar system bodies are neglected, harmonics in the planet gravitational potentials are neglected, solar radiation pressure is neglected, and the planet ephemerides are computed by means of osculating conics approximating the planet orbits. This simplified method of ephemeris generation also facilitates fast trajectory integration backward in time as well as forward, which is a very useful capability in state transition matrix computations.

This program, per se, is very useful for the intended special purpose. The imprecise solar system model, of course, renders it inapplicable to any type of study requiring a highly accurate computation of a real interplanetary trajectory. (Provision has been made, though, to modify it to use tape - stored precise planet ephemerides and to include the neglected influences should the need arise.) However, in the preparation of this program some rather novel trajectory computation techniques have been developed which individually have wider potential applicability to trajectory programs for other purposes. These special techniques are listed below.

1. Trajectory computation both forward and backward in time and the use of this capability for a "closed loop" accuracy check in the program.

2. Use of the state transition matrix to mechanize a simple and efficient trajectory search procedure.

3. A new formulation of Kepler's problem for hyperbolic conics for high precision computation, which achieves a very significant improvement in the computational accuracy of the conic position and velocity used in the Encke method.

These techniques are explained in considerable detail in Chapter III, since they may have an application to a wider range of trajectory computation problems. In the following section, Chapter II, a general description of the program is given. Methods and capabilities that are standard are only mentioned, but certain of the computational methods which are not standard and the special techniques listed above are described in detail in Chapter III. Chapter IV briefly describes the program functional flow and the functions of the individual subprograms. Detailed flow charts, data format descriptions, and program listings will be found in the companion document, the program Users' Manual (Reference 1).

The technical descriptions given in this document apply both to the single-precision and double-precision versions of the program. There are, of course, significant differences in the program structures for the two versions, but these are described in the Users' Manual (Reference 1).

CHAPTER II

GENERAL PROGRAM DESCRIPTION

2.1 Program Operating Characteristics

2.1.1 State and State Transition Matrix Integration

The program computes the state vector by integrating the many body equations of motion and the state transition matrix by integrating the state variational equations. The two integrations are done simultaneously, so that one pass of the program yields both the state and state transition matrix as required. The state transition matrix is, of course, state-dependent, and the way in which the two integrations are folded together is described in Sections 3.1 and 3.2. The state integration is done by Encke's method, and the numerical integration technique used is Nystrøm's method (Section 3.3). The integrations start with specified initial conditions (an initial state for the state integration and an identity matrix for the state transition matrix integration) and proceed until the specified time of flight has elapsed.

2.1.2 Reference Coordinates and Units

The reference coordinates for the computations are determined by reference planet ephemerides which are entered as input data. These ephemerides must always be expressed in heliocentric non-rotating, non-accelerating coordinates; ecliptic coordinates of a reference date (usually 1950.0) are almost always used in the studies in this laboratory, but heliocentric equatorial or other coordinates may be used equally well. These ephemeris reference coordinates then establish the directions of the reference coordinates of the program. However, the program reference coordinates are planetocentric or heliocentric in different phases of the trajectory, depending upon which of the bodies in the solar system exerts the primary influence determining that trajectory phase. The initial phase is specified in the program input data, and the initial state vector must be specified in the reference coordinates centered at the primary body of the initial phase. For example, if ecliptic reference coordinates

are used, for an Earth injection the initial phase is earth-centered and the initial state vector must be specified in Earth-centered ecliptic coordinates. After the initial point a phase determination is made at the end of each time step of the integration according to the test described in Section 3.6, and phase changes take place automatically when the test indicates the necessity. The phase change procedure is simple vector addition or subtraction of the reference body ephemerides and the spacecraft state. Since no coordinate rotation occurs at a phase change, no change is required in the state transition matrix.

Computational units used in the program are kilometers (distance), km/sec (velocity), and seconds (time). The reference planet ephemerides are entered in units of a.u. and a.u./day, because these units are used in the source of planet ephemerides. Conversion to km and km/sec is done in the program. The zero time reference in the program is immaterial, as long as the start time, end time, and epochs of the reference ephemerides specified in the input data all have a common zero time reference. For computational purpose the program sets the start time to zero and reckons all times from this reference.

2.1.3 Two-directional Integration Capability and Closed-loop Accuracy Check

As mentioned in the Introduction, the program has the capability of trajectory integration backward as well as forward in time. This capability facilitates economical computation of state transition matrices at several required points along a trajectory. For example, suppose that along an Earth-Mars transfer it is planned to make corrections at three points by means of a guidance technique using the integrated state transition matrix. Then the matrices relating target point miss to state deviations measured at each of the three points are required. With integration backward in time from Mars to Earth the nominal trajectory and all three required matrices can be generated directly by a single integration, while more complex computations are required if only integration forward in time is allowed.

This forward-backward integration capability provides the program with a rather unique "closed-loop" accuracy check. This accuracy check works as follows. A trajectory between a starting point and a target point is found by integration in, say, the forward time direction. Then a reverse integration is performed using the precise terminal state from the forward integration as the initial condition.

might be characterized as the fixed time of transfer, many-body Lambert problem. The search computations are linear, and, as explained in Section 3.7, the range of misses for which the linear search converges is as large as about half the distance of the target point from the Sun in cases tested to date. For example, if the target point is in the near vicinity of a target planet, if the miss on a search iteration is so large that the trajectory endpoint lies outside the sphere of influence, the search computations are done in heliocentric coordinates, and the range of linearity is then determined by the ratio of the miss and the heliocentric target point radius. Table 3-1 in Section 3.7 shows some typical examples illustrating the efficiency of the search procedure. The efficiency is relatively good, and this can be attributed primarily to the computation of a new state transition matrix automatically in each search iteration, so that the correct matrix is always available to be used in computing a correction to the starting velocity.

In the cases tested,the specified time of flight has always been large enough that the heliocentric transfer is elliptical with an eccentricity of the osculating ellipse significantly less than unity, and satisfactory convergence has been obtained. No tests have been made with near-parabolic transfer conditions.

The initial guess for the starting point velocity is not highly critical, except for a near 180° heliocentric transfer, because of the extended range of convergence mentioned above. However, if the initial guess is reasonably good, the search efficiency is further enhanced. A simplified, patched-conic procedure has been used successfully to generate first guess velocities, and this procedure is described in Section 3.7. The program itself may be used to generate conic orbits. The planet and Sun gravitational constants are included in the program input data; if any of these constants are set to zero, the effects of the corresponding bodies are "switched off".

2.1.6 Operating Modes and Printout

The program has three operating modes to which the data printout provisions are closely coupled.

2.1.6.1 Mode 1

Mode 1 is intended primarily for checkout purpose. A complete data printout occurs at the beginning point and the end of every time step (automatically included are the phase change points and the endpoint) in a single integration. The program will not search in

The reverse integration should reach the starting point with a terminal state that matches the original initial state for the first integration, if computational errors are negligible. The difference between the two states at the starting point indicates the computational inaccuracy. This capability has been found to be most useful in determining the effects of computational errors in the program.

2.1.4 Simplified Ephemeris Generation

The planet ephemerides are computed from osculating conics approximating the planet orbits. The elements of the osculating conics are determined from the reference ephemerides entered as input data. These reference ephemerides are both the positions and velocities of the five planets Venus, Earth, Mars, Jupiter, and Saturn at reference epochs chosen to give a good approximation to the planet orbits during the flight. In the experience of this group the most convenient source of these data has been the Jet Propulsion Laboratory Ephemeris Tapes E9510, E9511, and E9512, covering the years 1950 to 2000(reference 4). A special program has been written to convert the ephemerides on these tapes, which are in heliocentric equatorial coordinates of 1950.0, to heliocentric ecliptic coordinates of 1950.0 and list the data in a readable format. The reference ephemerides for the trajectory to be computed are taken directly from the listings and used as input data. Provision is made in the first operating mode (Mode 1) of the trajectory program to print the computed planet ephemerides at each time step in the computation. These printed data can be used to check the quality of the approximate ephemerides by comparison with the listed ephemerides.

2.1.5 Trajectory Search Capability

The program has a trajectory search capability. Given a starting position with respect to the initial body, a target point with respect to a terminal body, a fixed time of flight for the transfer, and an initial guess as to the starting velocity, the program will perform an iterative search for a trajectory by updating the starting velocity. A submatrix of the state transition matrix relates terminal point miss to initial point velocity deviations, and this relation is used to mechanize the iterative search. The search procedures are described in detail in Section 3.7.

The search attempts to satisfy three terminal position constraints by varying only the three initial velocity components, which

Mode 1; only a single trajectory computation is performed but the program does compute and print out the target point miss and the updated starting velocity for the next search iteration. In other words, the program cycles through a complete trajectory search iteration once, and the printout provides a trace of all computations performed.

The data printout at the end of each time step includes the following list of information (the Users' Manual, Reference 1, describes the record format):

- Elapsed time from start
- Primary body
- True state vector
- Encke conic state vector (see Section 3.1)
- Encke variable state vector (see Section 3.1)
- Disturbing acceleration defined in the Encke method
- Gravitational gradient matrices
  - due to the primary body (see Section 3.2)
  - due to the perturbing bodies (see Section 3.2)
- State transition matrix
- Product of state transition matrix and its inverse and the RMS Matrix, which are checks on transition matrix computational accuracy (optional, controlled by command in the input data; See Section 2.2.2.2)
- Planet positions

At each phase change point two printouts occur. The first is a complete printout as described above before the phase is changed, and the second repeats the above list after the phase is changed, except for the state transition matrix, the optional computational checks, and the planet positions, none of which varies across the phase change.

Mode 1 printout also includes a final record listing the results of the search computations to update the starting velocity for a new search iteration. The listed information in this final record includes the following:

- Updated starting velocity
- Matrix product NN^{-1} to check the updating computations (see Sections 3.2 and 3.7)
- Position miss vector components and magnitude

2.1.6.2 Mode 2

Mode 2 of the program is the usual search mode. In this mode the program automatically cycles through trajectory integrations, updating the starting velocity after each iteration according to the search computations described in Section 3.7. The input data include a maximum number of search iterations and a tolerance on the allowable miss of the target point. The search will terminate under program control either when a miss is obtained which is within the tolerance specified, or else when the maximum number of search iterations is reached.

Automatic printout in Mode 2 occurs only at the beginning point, the phase change points, and the endpoint of the trajectory. Furthermore, a control parameter in the input data determines whether this printout occurs in each search iteration or only in the final iteration. The printed data in each record include all the information in the first of the two lists above, except for the planet positions which are printed in Mode 1 only. In each iteration for which printout occurs, a final record lists the position miss vector components and magnitude.

It is also worth noting that Mode 2 can operate with the input data control determining the maximum number of search iterations set to 1 so that only one iteration occurs. This accomodates the cases in which limited printout it required along an established trajectory without either a search or the voluminous printout of Mode 1.

2.1.6.3 Mode 3

Mode 3 is also a search mode, differing from Mode 2 only in that the full state transition matrix is not computed. This mode is intended primarly to be used in preliminary trajectory searches in which the state transition matrix is of no use other than to mechanize the search and fast computation time is desired for search efficiency. Only a portion of the state transition matrix is required for the search computations, and only that portion is computed in Mode 3. The printout in Mode 3 is identical with Mode 2 except that the state transition matrix and the associated error check matrices are not printed.

2.1.6.4 Special Printout Provision

In addition to the automatic printout for each of the three operating modes, provision has been made in the program to obtain printouts at up to twenty specifiable time points along the trajectory. These special printouts can be obtained in any of the operating modes. The times of the special printouts are entered in the input data, and the times must be entered chronologically from the starting time of the

trajectory computation. The method which is used to compute the data at the special times is as follows. At the beginning of each normal time step in the program a test is made before the numerical integration commences to determine if the next required printout time lies within that time step. If it does not, the integration proceeds normally; if it does, the following special computational procedure occurs. The current time point in the integration becomes a point of demarcation, and all necessary current data for that point are specially stored. Then a special time step is computed from the present time to the required printout time, and a special integration step takes place, generating the data required for printout. After the printout occurs, the program restores the specially stored data at the point of demarcation, and the normal integration proceeds from that point with the next normal time step, and the printout time point test recommences with the next required printout time.

This method is somewhat more costly than a method which would simply inject an irregular time step at each printout point, and may be more costly than an interpolation method. However, it has the relative advantages that the overall integration is free of any noise resulting from irregular time steps and the printed data are free of interpolation errors.

2.1.7 Osculating Conic Data Option

The program includes an option for computing and printing the parameters and other data of the osculating conic approximating the trajectory at each printout time point. A control parameter in the input data determines whether or not the ØSCØN subroutine is called at each printout time. This subroutine computes and prints the following osculating conic data:

Conic type (ellipse, hyperbola).

Unit vectors locating the principal axes of osculating conic.

Unit vectors locating the ascending line of nodes and the transverse axis in the osculating orbit plane.

Unit vectors locating the R,S,Z coordinates (Reference 2).

Unit vectors locating the P,Q,Z coordinates (Reference 2).

Special angles γ and g (Reference 2).

Longitude of the node, argument of the pericenter, and the inclination angle of the osculating conic.

Semi-major axis, eccentricity, epoch of pericenter passage, true anomaly, hyperbolic anomaly, and mean anomaly (for a hyperbola).

Semi-major axis, eccentricity, epoch of pericenter passage, true anomaly, eccentric anomaly, and mean anomaly (for an ellipse).

It should be noted that the osculating conic described by these data is not the osculating conic used in the Encke integration method as described in 3.1. The latter conic is distinguished in the program terminology as the "Encke conic".

2.2 Computational Accuracy

2.2.1 Error Sources

Special measures have been necessary to limit computational inaccuracies to tolerable levels, especially in the single-precision version of the program. There are three categories of computational errors present in the program which are particularly sensitive in the single-precision computation. These are listed below and discussed in turn in the following paragraphs:

1. Truncation errors in the numerical integration and roundoff errors due to the limited precision (8 decimal figures in the IBM 7094) in normal computations.

2. Errors due to loss of precision at the phase change points and the rectification points in the Encke method.

3. Propagation of errors resulting from imprecision in the Kepler problem computations in the generation of the Encke conic state vector in the Encke method.

Double-precision computation, of course, greatly attenuates these error sources, with, however, an associated penalty in computer operating time.

2.2.1.1 Truncation and Roundoff Errors

Truncation errors in the numerical integration are kept small by use of a numerical integration method with a high order of accuracy and by proper choice of the time step for that method. The fourth-order Nystrøm method is used (Section 3.3); higher order Nystrøm methods are available, but the fourth-order method is sufficiently accurate. The time step computation is explained in detail in Section 3.5, with a derivation in Appendix B, and one basis of this computation is that the truncation error be less than the precision level, that is, less than 1 part in 10^8 for single-precision computation.

Truncation error is not the only determining factor in the time step computation. Roundoff errors which are due to the limited pre-

cision in the computations, tend to grow rapidly larger as the number of time steps is increased. So, there is in general, a tradeoff between truncation errors and roundoff errors in the time step computation. The time step formula contains a scaling constant, and there is a value of this scaling constant for which the total error from these two sources is minimized. This optimum value is furthermore trajectory-dependent, and should be redetermined in each case. The values given in Section 3.5 have been found to be a good first guess, generally.

2.2.1.2 Loss of Precision at Phase Changes and Encke Conic Rectifications

The errors which result from loss of precision at the phase change points and the rectification points in the Encke method(technically, roundoff errors) are unfortunate and, generally, irreducable consequences of single-precision computation. To illustrate the nature of this error source, consider a trajectory leaving Earth, going, say, to Mars. At the first phase change point from geocentric to heliocentric coordinates the distance from earth is of the order of 10^6km, and the distance of the earth from the Sun is of the order of 10^8km. Therefore, in the phase change computations in the computer, numbers of the following formats must be added:

.XXXXXXXX E 07	distance of point from Earth
.YYYYYYYY E 09	distance of Earth from Sun

When these two are added, the least significant two digits of the first are lost. In the present example this is a position error of between 0 and 10 km. A similar error, but of quite different magnitude, is made also in each velocity component. When these errors are propagated forward to the target point, they result in a miss. The position and velocity errors resulting from the phase change are not independent. Furthermore, in this example there is at least one more phase change at the Martian sphere of influence. Consequently, a general relation which predicts the target miss due to these error sources is not easily derived.

At each rectification point in the Encke method (see Section 3.1) errors of precisely the same nature occur. When rectification occurs, the numerically integrated deviation is two orders of magnitude smaller than the total state vector, so a loss of precision occurs in the rectification and a miss results. On a trajectory such as the one in the example above, typically four rectifications occur, one at each of

the phase change points (required by the change of primary body and conic type), one in the midcourse phase, and one in the Mars approach phase.

Double-precision computation is the only effective method of reducing errors from these sources.

2.2.1.3 Errors in the Hyperbolic Encke Conic Computation

This error source was most troublesome in the development of this program, and a special formulation of Kepler's problem was necessary to reduce errors of this type to tolerable levels. Section 3.4 and Appendix A describe the nature and consequences of this error source and the special formulation devised to control the errors. Very briefly, the equations of standard formulations of the Kepler problem are critically sensitive to computational precision errors in the Kepler iteration variable for the hyperbolic phases of trajectories in the vicinity of planets. These errors propagate exponentially in the hyperbolic equations and result both in large misses and in frequent failures of the Kepler iteration variable to converge to within an acceptable tolerance. It was found that it was not possible to compute the conic state to single-precision accuracy with any of the normal conic formulations tested (two varieties of the standard conic formulation and the Battin universal formulation). It has been found possible to obtain this accuracy with single-precision computation using the new formulation, and this is regarded as an accomplishment which may be of significant importance in other Encke method programs.

2.2.2 Computational Accuracy Checks

The program provides three means of checking computational accuracy: (1) the search noise level, (2) accuracy checks on the state transition matrix computation, and (3) the "closed loop" accuracy check. These are discussed in turn below.

2.2.2.1 Search Noise Level

The search noise level is determined after a representative nominal trajectory has been computed by setting the target miss tolerance in the input data to zero and allowing the program to cycle through a sufficient number (usually 6 to 12) of search iterations. It is understood that the nominal trajectory is within the noise band: that is, there is no bias in the miss to be removed by further searching when the noise level determination is made.

The search noise level is an indication of random computational errors. There are possible systematic errors in the single-precision computations, particularly for very small perturbations about the nominal trajectory, and these are not revealed in the search noise level. In the cases tested thus far, the target miss noise level reflects an uncertainty in the initial velocity of less than 1 part in 10^7 (single-precision), if the time step scaling factor is properly chosen. That is, if one of the initial velocity components for the nominal trajectory is changed by 1 part in 10^7, the resulting miss will be well above the noise level, unless the time step scaling factor is not properly chosen. This serves, then, as one means of determining a proper time step scaling constant; the value is optimum when the noise level is minimum.

2.2.2.2 State Transition Matrix Accuracy Checks

There are two accuracy checks on the state transition matrix computations, which are optional depending upon a control constant in the input data. The first is the computed product of the state transition matrix C and its inverse C^{-1}, and the second is an "RMS" matrix to be described below. The product matrix CC^{-1} should be the (6x6) identity matrix, but, because of roundoff errors in the matrix multiplication, very large errors result in the product matrix. Consequently, this first accuracy check alone has limited usefulness; the accuracy of the state transition matrix cannot be easily inferred from the product matrix. The RMS matrix, however, can be used in conjunction with the CC^{-1} matrix to infer an order of magnitude of the state transition matrix term errors. The paragraphs below give a very brief description of this method, and Figure 2-1 depicts a state transition matrix and its associated product and RMS matrices used to illustrate the discussion. Appendix C gives a detailed derivation of the effects of errors in the CC^{-1} and RMS matrices.

The state transition matrix C may be partitioned into four (3x3) submatrices:

$$C = \begin{bmatrix} M & N \\ S & T \end{bmatrix} \tag{2.1}$$

Then, since C is simpletic, the inverse is obtained simply by rearrangement of terms:

$$C = \begin{bmatrix} T^T & -N^T \\ -S^T & M^T \end{bmatrix} \tag{2.2}$$

STATE TRANSITION MATRIX (C)

-.14126371E 04	.27839328E 04	-.82402013E 03	-.22570312E 08	-.11218049E 08	.22714179E 07
.21399605E 02	.23297542E 04	-.93005955E 03	-.57208001E 07	-.12805637E 08	.14105091E 07
-.34655884E 02	-.74310082E 03	-.84448577E 03	.78804231E 06	.10795759E 07	-.64414939E 07
.25262041E-03	-.49001477E-03	.13997025E-03	.40127818E 01	.19517754E 01	-.42379021E 00
-.69631811E-05	-.42409994E-04	.18672670E-04	.33267700E-01	.25323221E 00	-.21051847E-01
-.48797327E-05	.17993081E-04	.69163419E-05	-.91190971E-01	-.49712444E-01	.80609781E-01

PRODUCT OF STATE TRANSITION MATRIX AND ITS INVERSE (CC^{-1})

.99961090E 00	.13775826E-02	-.15521049E-03	-.22400000E 03	.58080000E 05	-.13808000E 05
-.10616302E-01	.10002470E 01	.17786026E-03	-.58656000E 05	-.38400000E 03	-.17440000E 04
.24871826E-02	-.12397766E-04	.99996185E 00	.14016000E 05	.19200000E 04	.63999999E 02
.00000000E 00	-.24681412E-09	.28478553E-10	.99969101E 00	-.10402679E-01	.24528503E-02
.24965630E-09	.12434498E-12	-.49222848E-11	.14042854E-02	.10002713E 01	-.16689301E-04
-.28649083E-10	.48032689E-11	-.00000000E 00	-.15544891E-03	.17547607E-03	.99996090E 00

RMS MATRIX

.89548803E-04	.15882070E-05	-.50819226E-06	-.35458289E-08	.12905599E-05	-.13034038E-05
-.13445691E-05	.12448058E-02	.65827718E-06	-.13033589E-05	-.90924690E-08	-.15053056E-06
.13521736E-05	-.54200453E-07	.10919808E-01	.13230379E-05	.16572171E-06	.82301645E-08
.00000000E 00	-.16213911E-05	.53096886E-06	.89555980E-04	-.13175135E-05	.13335086E-05
.16400621E-05	.81796461E-08	-.93151677E-06	.16189926E-05	.12448361E-02	-.72962148E-07
-.53414831E-06	.90899364E-06	-.00000000E 00	-.50897290E-06	.64945310E-06	.10919798E-01

Fig. 2.1 State Transition Matrix and Error Check Matrices for a Typical Example

The CC^{-1} product matrix is obtained by straightforward multiplication:

$$CC^{-1} = \begin{bmatrix} (MT^T - NS^T) & (-MN^T + NM^T) \\ (ST^T - TS^T) & (-SN^T + TM^T) \end{bmatrix} \qquad (2.3)$$

The computations in the program follow this pattern precisely, and from Equation (2.3) two observations can be made immediately:

1. The diagonal elements of the upper right and lower left(3x3) submatrices of CC^{-1} should be identically zero regardless of the errors in the terms of C. This is because the six multiplied terms in the computation of each of these particular diagonal elements consist of three pairs of identically equal and opposite terms. The non-zero diagonal terms in these two submatrices then result totally from roundoff errors in the multiplications and additions involved in the computations. Reference to Figure 2-1 shows that these errors can be very large in the upper right (3x3) submatrix especially.
2. The upper left (3x3) submatrix should be the identical transpose of the lower right (3x3) submatrix regardless of the errors in the terms of C. If the transpose property is not identically true (as is the case in Figure 2-1), the reason is that the roundoff errors in the computation of corresponding terms sum together in different orders in the two computations. It is also true that in these two submatrices the term value differences from 1 (diagonal terms) or 0 (off-diagonal terms) are due to both roundoff errors in the matrix product computations and inaccuracies in the state transition matrix.

Since some of the terms in the CC^{-1} matrix result from matrix product roundoff errors only and some from both roundoff errors and transition matrix term errors, it is possible to ascertain the order of magnitude of the roundoff errors only, the order of magnitude of the combination errors, and then from a comparison infer a limit for the order of magnitude of the transition matrix errors. The RMS matrix is intended to facilitate this operation. The method of forming the RMS matrix is as follows. Consider the ijth element of CC^{-1}:

$$(CC^{-1})_{ij} = \sum_{k=1}^{6} (C)_{ik} (C^{-1})_{kj} \qquad (2.4)$$

The ijth element of RMS is $(CC^{-1})_{ij}$ divided by the square root of the sum of the squares of the six terms in the computation of $(CC^{-1})_{ij}$:

$$(RMS)_{ij} = \frac{(CC^{-1})_{ij}}{\sqrt{\sum_{k=1}^{6} \left[(C)_{ik} \quad (C^{-1})_{kj}\right]^2}} \tag{2.5}$$

An analysis of the propagation of both roundoff errs and state transition matrix inaccuracies in the computed CC^{-1} and RMS matrices is given in Appendix C. There are two assumptions made in the course of analysis which lead to a simplified result:

1. The relative roundoff errors in all multiplications are independent and unbiased and have the same distribution with a common variance σ^2_{rel}. (Relative error is the absolute term error divided by the term value.)
2. The relative state transition matrix term errors are also independent and unbiased and have a common distribution with a variance σ^2_t

It is shown in Appendix C with the aid of these assumptions that the order of magnitude of the variance σ^2_{rel} of the relative roundoff errors can be determined from

$$\sigma^2_{rel} \sim \sigma^2_{(RMS)_{ij}} \tag{2.6}$$

where i and j denote the diagonal terms of the upper right and lower left (3 x 3) submatrices of RMS. That is, ij has the following permissable values

$$ij = 14, 25, 36, 41, 52, 63$$

$\sigma^2_{(RMS)_{ij}}$ is the variance of the $(RMS)_{ij}$ term, and can be estimated from the printed data. The order of magnitude of the variance of the relative state transition matrix term errors can be obtained from

$$\sigma^2_{rel} + 2\,\sigma^2_t \sim \sigma^2_{(RMS)_{kl}} \tag{2.7}$$

where kl has any value other than the six values of ij above.

The procedure for using the error checks is as follows:

1. Determine the order of magnitude of the variance of the relative roundoff errors from the diagonal elements of the upper right and lower left (3 x 3) submatrices of the RMS matrix, using Equation (2.6).

2. Next, determine the order of magnitude of the variances from other elements of the RMS matrix using Equation (2.7).

3. Compare the results of steps 1 and 2. If the order of magnitude in step 2 is about the same as in step 1, the relative state transition matrix term errors are no larger than the relative roundoff errors. If the result of step 2 is significantly larger than step 1, then the relative state transition matrix term errors are significant and the order of magnitude of the variance σ_t^2 can be determined from Equation (2.7)

Consider Figure 2-1 again for an illustration. In applying step 1 we use all six of the special elements of RMS to obtain an estimate of σ_{rel}^2 :

$$\sigma_{rel}^2 \sim \frac{1}{6} \left[(RMS)_{14}^2 + (RMS)_{25}^2 + (RMS)_{36}^2 + (RMS)_{41}^2 + (RMS)_{52}^2 + (RMS)_{63}^2 \right]$$

$$\sim .38 \times 10^{-16}$$

In applying step 2 we do not use the main diagonal elements of RMS, because they are nonuniformly biased. We arbitrarily choose the six off-diagonal elements of the upper left (3 x 3) submatrix:

$$\sigma_{rel}^2 + 2\sigma_t^2 \sim \frac{1}{6} \left[(RMS)_{12}^2 + (RMS)_{13}^2 + (RMS)_{21}^2 + (RMS)_{23}^2 + (RMS)_{31}^2 + (RMS)_{32}^2 \right]$$

$$\sim 1.14 \times 10^{-12}$$

We conclude, then, that the relative term errors are much larger than the relative roundoff errors, and

$$\sigma_t \sim .5 \times 10^{-6}$$

2.2.2.3 Closed-loop Accuracy Check

The "closed" loop accuracy check evaluates the effects of both random and systematic errors in the trajectory integration. It is a most powerful overall accuracy determination. The mechanization of the closed loop accuracy check has been explained earlier in Section 2.1.3. The closure miss in this check is the net result of the contributions from all error sources, and the individual contributions are not separable, in general. This powerful technique has potential applicability to other trajectory computation programs for different purposes.

2.2.3 Typical Accuracy Data

Table 2-1 displays some typical data from single-precision program accuracy tests. The closure miss data are the results of closed-loop accuracy checks for the different trajectories, and the search

noise level in each case is for a trajectory search in the forward direction. Comparison of the first two examples indicates the error contribution of a phase change, and the last example shows the improvement obtained by optimizing the time step scaling factor. Example 3 shows typical accuracy of the program for a full interplanetary trajectory with all error sources contributing in full measure to the results.

TABLE 2-1

PROGRAM ACCURACY DATA

(Single Precision Computations)

Trajectory	RMS Search Noise Level	Closure Miss
1. Point just inside Martian sphere of influence to a target point 5000 km above planet surface (trajectory search and closed-loop check).	0.2km	5.4km
2. Point on same trajectory just outside Martian sphere of influence to same target point (search and closed-loop check).	3	105
3. Earth-Mars 258 day transfer, target point 5000 km above planet surface (search and closed-loop check)	65	5466
4. Midcourse point 120 days from Earth to same target point at Mars for the above 258 day trajectory (search and closed-loop check for different time step scaling factors)		
SF = .050	10.3	1084
SF = .100	14.0	269

CHAPTER III

COMPUTATION METHODS AND TECHNIQUES

The computation methods and techniques used in the program will be discussed in some detail in this section. The discussion will point out some acute problems associated with various techniques that have been attempted.

3.1 State Intergration Method

Reference 3 describes the Encke method as it is used in this program. The fundamental vector equation of motion is

$$\frac{d^2\underline{r}}{dt^2} + \frac{\mu_{pri}}{r^3}\,\underline{r} = -\sum_{k=1}^{n} \mu_k \left(\frac{\underline{d}_k}{d_k^3} + \frac{\underline{r}_k}{r_k^3} \right) = \underline{a}_d \qquad (3.1)$$

where: $\underline{r}$ is the true position of the spacecraft with respect to the primary body.

$\mu_{pri} = GM_{pri}$ is the product of the universal gravitational constant and the mass of the primary body (the spacecraft mass is neglected with respect to the masses of the planets).

The summation is over all bodies other than the primary body.

μ_k is the product of universal gravitational constant and the mass of the kth body.

$\underline{d}_k$ is the spacecraft position with respect to the kth body.

$\underline{r}_k$ is the position of the kth body with respect to the primary body.

The primary body is defined to be the one which exerts the largest gravitational force on the spacecraft at any point on the trajectory: that is, the spacecraft flies primarily under the gravitational influence of the primary body, and the gravitational attractions of the remaining bodies constitute a disturbing force which causes the tra-

jectory to be perturbed from a simple two-body trajectory. The primary body changes during the course of the trajectory, and the "phase" of the trajectory is identified by the primary body. For example, in the injection phase the primary body is the launch planet; in the mid-course phase it is the Sun; and in the terminal phase it is the target planet. The method of phase determination and changing in the program is discussed in Section 3.6.

In the Encke method at a time t_0 (known as a rectification time) the true position and velocity vectors $\underline{r}(t_0)$ and $\underline{v}(t_0)$ define an osculating conic orbit. In the program terminology this is called the Encke conic to differentiate it from an osculating conic which the program can compute at any time point if an input control command is given. The Encke conic is the solution to the homogeneous portion of Equation (3.1):

$$\frac{d^2\underline{r}_{enc}}{dt^2} + \frac{\mu_{pri}}{r^3_{enc}}\underline{r}_{enc} = 0 \qquad (3.2)$$

The Encke conic computation is done in the ENCØN subroutine. At a time t the true velocity and position are the sum of the Encke conic velocity and position and the increments caused by the disturbing forces:

$$\begin{aligned} \underline{r}(t) &= \underline{r}_{enc}(t) + \underline{\delta}(t) \\ \underline{v}(t) &= \underline{v}_{enc}(t) + \underline{\nu}(t) \end{aligned} \qquad (3.3)$$

The differential equation of $\underline{\delta}(t)$ is derived from (3.1) using (3.2) and (3.3):

$$\frac{d^2\underline{\delta}}{dt^2} = \frac{\mu}{r^3_{enc}}\left[\left(1 - \frac{r^3_{enc}}{r^3}\right)\underline{r} - \underline{\delta}\right] + \underline{a}_d \qquad (3.4)$$

This is the basic trajectory equation integrated numerically in the INTEG subroutine. Both terms on the right side of (3.4) are computed in the GRAVFØ subroutine. Both subroutines perform additional compu-

tations for the state transition matrix integration, as described below.

3.2 State Transition Matrix Integration Method

The derivation of the state transition matrix and its applications to simplified interplanetary guidance are described in detail in reference 2. The matrix relates the propagation of small perturbations in the state vector along a reference trajectory according to the equation:

$$\delta\underline{x}_j = C_{ji}\,\delta\underline{x}_i \tag{3.5}$$

where

$\underline{x}(t) = \begin{bmatrix} \underline{r}(t) \\ \underline{v}(t) \end{bmatrix}$ is on the state vector at time t on a reference trajectory

$\delta\underline{x}_i \quad \begin{bmatrix} \delta\underline{r}(t_i) \\ \delta\underline{v}(t_i) \end{bmatrix}$ is the state error at t_i

$\delta\underline{x}_j$ is the state error at t_j

C_{ji} is a (6x6) matrix relating $\delta\underline{x}_j$ to $\delta\underline{x}_i$

If the state errors are small, the linearized state transition matrix is

$$C_{ji} = \left\| \frac{\partial \underline{x}_j}{\partial \underline{x}_i} \right\| \tag{3.6}$$

C_{ji} is partitioned into four (3x3) submatrices in the following manner

$$C_{ji} = \begin{bmatrix} M_{ji} & N_{ji} \\ S_{ji} & T_{ji} \end{bmatrix} = \begin{bmatrix} \left\| \frac{\partial \underline{r}_j}{\partial \underline{r}_i} \right\| & \left\| \frac{\partial \underline{r}_j}{\partial \underline{v}_i} \right\| \\ \left\| \frac{\partial \underline{v}_j}{\partial \underline{r}_i} \right\| & \left\| \frac{\partial \underline{v}_j}{\partial \underline{v}_i} \right\| \end{bmatrix} \tag{3.7}$$

and it is further shown in Reference 2 that C_{ji} obeys the following differential equation:

$$\frac{dC_{ji}}{dt_j} = \begin{bmatrix} \frac{dM_{ji}}{dt_j} & \frac{dN_{ji}}{dt_j} \\ \frac{dS_{ji}}{dt_j} & \frac{dT_{ji}}{dt_j} \end{bmatrix} = \begin{bmatrix} S_{ji} & Tji \\ G_j M_{ji} & G_j N_{ji} \end{bmatrix} \tag{3.8}$$

where the (3x3) matrix G_j is the gravitational gradient matrix defined by the position variational equation

$$\delta\ddot{\underline{r}}(t_j) = G_j \delta\underline{r}(t_j) \tag{3.9}$$

Note that t_i is fixed and t_j varies, and C_{ji} is a function of both t_i and t_j. G is recognized to consist of two parts, the gravitational gradient of the primary body, denoted by G_{pri}, and the resultant gravitational gradient of the disturbing bodies, denoted by G_{pert}. Then

$$G_j = G_{pri}(t_j) + G_{pert}(t_j) \tag{3.10}$$

where

$$G_{pri}(t_j) = \frac{\mu_{pri}}{r(t_j)^5} \left[3\underline{r}(t_j)\underline{r}(t_j)^T - \underline{r}(t_j)^T\underline{r}(t_j) I_3 \right] \tag{3.11}$$

I_3 is the (3x3) identity matrix

$$G_{pert}(t_j) = \sum_{k=1}^{n} \frac{\mu_k}{d_k(t_j)^5} \left[3\underline{d}_k(t_j)\underline{d}_k(t_j)^T - \underline{d}_k(t_j)^T\underline{d}_k(t_j) I_3 \right] \tag{3.12}$$

where the summation extends over all disturbing bodies and $\underline{d}_k$ is the spacecraft position with respect to the kth body.

Equation (3.8) is the basic matrix differential equation which is integrated in the INTEG subroutine. The G matrix is computed in the GRAVFØ subroutine. The initial condition from which the integration begins is

$$C_{ii} = I_6, \text{ the (6x6) identity matrix} \tag{3.13}$$

3.3 Numerical Integration Technique

Nystrøm's method of numerical integration is used in the program. This technique is described in Reference 5. It is a one-step integration method which applies to second order differential equations of the special form

$$\frac{d^2X}{dt^2} = f(X) \tag{3.14}$$

where X may be a scalar, vector, or matrix. The state equation (3.4) is clearly of this form, and the state transition matrix equation (3.8) can be easily converted to the same form as follows. Define

$$C_{ji} = \begin{bmatrix} A_{ji} \\ B_{ji} \end{bmatrix} \tag{3.15}$$

where

$$A_{ji} = \begin{bmatrix} M_{ji} & N_{ji} \end{bmatrix} \quad (3x6)$$

$$B_{ji} = \begin{bmatrix} S_{ji} & T_{ji} \end{bmatrix} \quad (3x6)$$

When these are substituted into (3.8), the result is

$$\frac{dC_{ji}}{dt_j} = \begin{bmatrix} \frac{dA_{ji}}{dt_j} \\ \\ \frac{dB_{ji}}{dt_j} \end{bmatrix} = \begin{bmatrix} B_{ji} \\ \\ G_j A_{ji} \end{bmatrix}$$

Therefore

$$\frac{d^2A_{ji}}{dt_j^2} = G_j A_{ji} \tag{3.16}$$

Equation (3.16) is of the form (3.14).

The Nystrøm integration equations are given below for the general form (3.14). Let

$$Y = \frac{dX}{dt}$$

In the integration of the state equation (3.4) X corresponds to $\underline{\delta}$ and Y corresponds to $\underline{v}$. In the integration of the state transition matrix

equation (3.8) or (3.16) X corresponds to A_{ji} and Y corresponds to B_{ji}. In this terminology the Nystrøm equations are expressed as follows:

$$X_{n+1} = X_n + h\Phi(X_n, Y_n, h)$$
$$Y_{n+1} = Y_n + h\Psi(X_n, Y_n, h) \qquad (3.17)$$

where

$$\Phi(X_n, Y_n, h) = Y_n + \frac{h}{6}(k_1 + 2k_2) \qquad (3.18)$$

$$\Psi(X_n, Y_n, h) = \frac{1}{6}(k_1 + 4k_2 + k_3) \qquad (3.19)$$

$$k_1 = f(X_n) \qquad (3.20)$$

$$k_2 = f(X_n + \frac{h}{2} Y_n + \frac{h^2}{8} k_1) \qquad (3.21)$$

$$k_3 = f(X_n + hY_n + \frac{h^2}{2} k_2) \qquad (3.22)$$

$h = t_{n+1} - t_n$ is the time step increment

The integrations of both (3.4) and (3.16) are performed in the INTEG subroutine. The forcing function equations (3.20), (3.21), and (3.22) are evaluated with the use of three other subroutines, however. The right side of Equation (3.4)(the perturbing acceleration acting on the spacecraft) and the gravitational gradient matrix G in equation (3.16) are computed in the GRAVFØ subroutine. The Encke conic position and velocity vectors required for these computations are generated in the ENCØN subroutine, and the planet positions are computed in the EPHEM subroutine. These computations take place, of course, once during each time step in the integration. When k_1, k_2, and k_3 have been computed then Equations (3.17) are evaluated using (3.18) and (3.19).

The Nystrøm technique is self-starting and achieves a high order of accuracy with computational simplicity. The formulation used in the program yields results with truncation errors proportional to h^4; higher precision Nystrøm formulations are available(Reference 5).

3.4 Conic Computation Method

The most difficult problem experienced in the preparation of this

program has been the accurate computation of the conic position and velocity $\underline{r}_{enc}$ and $\underline{v}_{enc}$ used in the Encke integration method. This result may perhaps be surprising, since computation of conic orbits is generally considered to be straightforward. However, the conic equations are critically sensitive to computational imprecision in the case of a hyperbolic orbit in the vicinity of a planet. The conic equations are not so sensitive in the case of an elliptical orbit, and any of the usual formulations of Kepler's problem for an elliptical orbit can yield acceptable computational accuracy. Battin's universal formulation (Reference 3) also exhibits the critical sensitivity to computational imprecision in the case of hyperbolic orbits. Both the Battin formulation and the standard conic formulation were tried unsuccessfully in the program.

There are two deleterious effects of computational imprecision in the conic equations for hyperbolic orbits: (1) the iterative solution of Kepler's equation fails to converge to within an acceptable tolerance, and (2) the error in the iteration variable propagates exponentially in the computation of $\underline{r}_{enc}$ and $\underline{v}_{enc}$. These effects result in intolerable errors in single-precision trajectory computation. A new formulation of the hyperbolic conic equations has been devised to achieve high precision computation. This new formulation consists of both an advantageous change of variable in Kepler's equation and a careful grouping of terms in the equations to maximize computational precision. The new formulation has completely overcome the convergence failure in Kepler's equation and has achieved a very significant improvement in the propagation of errors in the iteration variable.

The discussion below will describe the precision problem for the standard hyperbolic formulation. The same argument applies generally to the Battin universal formulation. The new formulation will be described, and error propagations in the two formulations will then be compared.

The computation of $\underline{r}_{enc}$ and $\underline{v}_{enc}$ is, of course, Kepler's problem: given $\underline{r}_0$ and $\underline{v}_0$ at a time t_0 with respect to a primary gravitating body, determine the parameters of the conic orbit and compute the position and velocity $\underline{r}_{enc}(t)$ and $\underline{v}_{enc}(t)$ at a later time t. Figure 3.1 illustrates the problem for a hyperbolic orbit. The Kepler equation for this case has the form

$$\sqrt{\frac{\mu}{a^3}}\,\Delta t = -\Delta H + \frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}}\left[\cosh \Delta H - 1\right] + \left(1 + \frac{r_0}{a}\right) \sinh \Delta H \qquad (3.23)$$

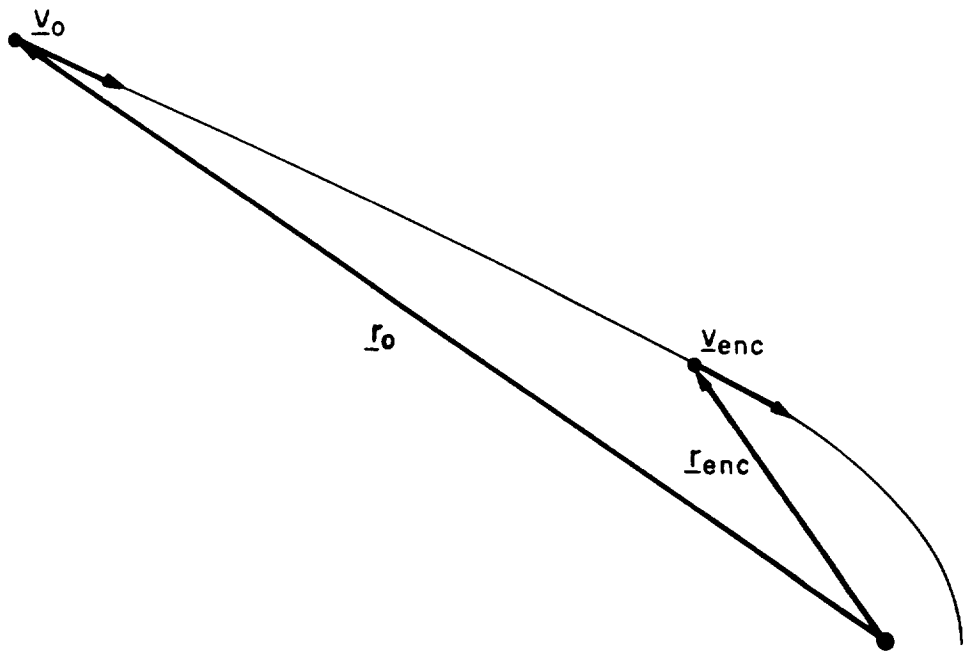


Figure 3.1 Kepler's problem for hyperbolic orbit

where

μ is the gravitational constant of the primary body

a is the semimajor axis of the hyperbolic orbit (positive)

$\Delta t = t - t_0$ is the known time increment

$\Delta H = H - H_0$ is the incremental change of the hyperbolic anomaly

$r_0 = \text{magnitude} \left| \underline{r}_0 \right|$

ΔH is determined by an iterative solution of this equation, and then $\underline{r}_{enc}(t)$ and $\underline{v}_{enc}(t)$ are determined from

$$\underline{r}_{enc} = \left[1 - \frac{a}{r_0}(\cosh \Delta H - 1) \right] \underline{r}_0 + \left[\frac{a}{\mu} \underline{r}_0 \cdot \underline{v}_0 \, (\cosh \Delta H - 1) + r_0 \sqrt{\frac{a}{\mu}} \sinh \Delta H \right] \underline{v}_0 \tag{3.24}$$

$$\underline{v}_{enc} = - \left[\frac{\sqrt{\mu a}}{r r_0} \sinh \Delta H \right] \underline{r}_0 + \left[1 - \frac{a}{r} (\cosh \Delta H - 1) \right] \underline{v}_0 \tag{3.25}$$

As a trajectory approaches a planet, $\underline{r}_0$ and $\underline{v}_0$ are first defined at the planet sphere of influence, and the scalar product $\underline{r}_0 \cdot \underline{v}_0$ is a large negative number. For the case of a close pass of the planet, the semimajor axis a will be of the order of a planet radius. Typical values of r_0 at the sphere of influence are more than 100 planet radii. Hence the coefficient $(1 + \frac{r_0}{a})$ in (3.23) is of the order of 100. The coefficient $\frac{\underline{r}_0 \cdot \underline{v}_0}{\mu a}$ is of the same order of magnitude since

$$\left(1 + \frac{r_0}{a} \right)^2 - \left(\frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}} \right)^2 = e^2 = 1 + \frac{\left| \underline{r}_0 \times \underline{v}_0 \right|^2}{\mu a} \tag{3.26}$$

where e is the eccentricity of the hyperbola with typical values of about 2. Consequently, when ΔH is of a reasonable size, the last two terms of (3.23) are large, of the same order of magnitude, and opposite in sign, and single precision computation results in both appreciable errors in ΔH and nonconvergence in the iteration.

On the IBM 7094, which is limited to 8 decimal digit single precision numbers, the tightest usable tolerance in a ratio test in the ΔH iteration is about 1×10^{-7}. Our experience has been that a Newton iteration for ΔH failed to converge for values of ΔH larger than about 2 for a case with the typical numbers mentioned above.

The convergence problem is not experienced on the outbound portion of the hyperbola because both of the large terms in (3.23) are positive. This formulation does, however, suffer the other unfortunate disadvantage that an error in ΔH propagates exponentially in $\underline{r}_{enc}(t)$ and $\underline{v}_{enc}(t)$. Differentiating (3.24) and (3.25) with respect to ΔH shows this property:

$$\delta\underline{r}_{enc} = -\left[\frac{a}{r_0}\sinh\Delta H\right]\delta(\Delta H)\underline{r}_0 + \left[\frac{a}{\mu}\underline{r}_0\cdot\underline{v}_0\sinh\Delta H + r_0\sqrt{\frac{a}{\mu}}\cosh\Delta H\right]\delta(\Delta H)\underline{v}_0 \tag{3.27}$$

$$\delta\underline{v}_{enc} = -\frac{\sqrt{\mu a}}{rr_0}\cosh\Delta H\ \delta(\Delta H)\underline{r}_0 - \frac{a}{r}\sinh\Delta H\,\delta(\Delta H)\underline{v}_0 - \frac{1}{r}\frac{dr}{d(\Delta H)}\delta(\Delta H)\left(\underline{v}_{enc} - \underline{v}_0\right) \tag{3.28}$$

All coefficients in these equations contain exponential multipliers which rapidly increase the magnitudes of the errors as the transfer lengthens.

The new formulation overcomes the iteration convergence problem and achieves a very significant improvement in error propagation. The formulation is derived by a change of variables. Define a new variable x by the relation

$$e^{\Delta H} = 1 + x \qquad (\Delta H > 0) \tag{3.29}$$

Kepler's equation can be put into the following two forms by substitution of (3.29) into (3.23) and algebraic rearrangement of terms:

$$\frac{K_1}{2}x^2 + K_2 x - t_m - (1+x)\log(1+x) = 0 \qquad (x \geq 0.3) \tag{3.30}$$

$$\frac{(K_1 - 1)}{2}x^2 + (K_2 - 1)x - t_m + x^3 L(x) = 0 \qquad (x < 0.3) \tag{3.31}$$

where

$$K_1 = \frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}} + \left(1 + \frac{r_0}{a}\right) = \frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}} + \left(\frac{r_0 v_0^2}{\mu} - 1\right) \quad (\underline{r}_0 \cdot \underline{v}_0 > 0) \tag{3.32}$$

$$= - \frac{1 + \frac{|\underline{r}_0 \times \underline{v}_0|^2}{\mu a}}{\frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}} - \left(\frac{r_0 v_0^2}{\mu} - 1\right)} \quad (\underline{r}_0 \cdot \underline{v}_0 < 0) \tag{3.33}$$

$$K_2 = \frac{r_0 v_0^2}{\mu} - 1 - t_m \tag{3.34}$$

$$t_m = \sqrt{\frac{\mu}{a^3}}\, \Delta t \tag{3.35}$$

$$\frac{1}{a} = \frac{v_0^2}{\mu} - \frac{2}{r_0} \tag{3.36}$$

$$L(x) = \frac{1}{x^3} \left\{ \frac{x^2}{2} + x - (1+x) \log(1+x) \right\} \tag{3.37}$$

$$= \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{(n+2)(n+3)}$$

A detailed derivation of these equations is given in Appendix A.

Equation (3.31) is used for very short transfers ($x < 0.3$ corresponds approximately to $\Delta H < 0.25$), and (3.30) is used for all but very short transfers. The terms in these two equations have been especially grouped to maximize computational precision. The constant K_1 in (3.30) must be computed precisely, since it has a predominant effect for large transfers. Consequently, the alternative forms in equations (3.32) and (3.33) are used. The L(x) function in Equation (3.31) is evaluated by the power series expansion (3.37), the series being truncated when the sum in the computer is unchanged by the addition of the next term i.e., when computational precision is exceeded.

The conic state equations are derived by substituting (3.29) into (3.24) and (3.25) with the results (see Appendix A):

$$\underline{r}_{enc}(t) = \left[1 - \frac{a}{2r_0}\left(\frac{x^2}{1+x}\right)\right]\underline{r}_0 + \left[\frac{a}{2\mu}\,\underline{r}_0\cdot\underline{v}_0\left(\frac{x^2}{1+x}\right) + r_0\sqrt{\frac{a}{\mu}}\,\frac{x(1+\frac{x}{2})}{(1+x)}\right]\underline{v}_0 \tag{3.38}$$

$$\underline{v}_{enc}(t) = -\frac{\sqrt{\mu a}}{r r_0}\,\frac{x(1+\frac{x}{2})}{1+x}\,\underline{r}_0 + \left[1 - \frac{a}{2r}\left(\frac{x^2}{1+x}\right)\right]\underline{v}_0 \tag{3.39}$$

The effect of an error in x is

$$\delta\underline{r}_{enc} = -\frac{a}{2r_0}\,\frac{2x+x^2}{(1+x)^2}\,\delta x\,\underline{r}_0 + \left[\frac{a}{2\mu}\,\underline{r}_0\cdot\underline{v}_0\,\frac{2x+x^2}{(1+x)^2} + r_0\sqrt{\frac{a}{\mu}}\,\frac{1+x+\frac{x^2}{2}}{(1+x)^2}\right]\delta x\,\underline{v}_0 \tag{3.40}$$

$$\delta\underline{v}_{enc} = -\frac{\sqrt{\mu a}}{r r_0}\,\frac{1+x+\frac{x^2}{2}}{(1+x)^2}\,\delta x\,\underline{r}_0 - \frac{a}{2r}\,\frac{2x+x^2}{(1+x)^2}\,\delta x\,\underline{v}_0 - \frac{1}{r}\frac{dr}{dx}\,\delta x(\underline{v}-\underline{v}_0) \tag{3.41}$$

A comparison of (3.27) and (3.28) with (3.40) and (3.41) verifies that the latter equations show an improvement in the propagation of errors in the iteration variables. The errors $\delta(\Delta H)$ and δx are random variables and may be considered to be uniformly distributed within the range of the iteration tolerance, i.e.,

$$0 \le \frac{\delta(\Delta H)}{\Delta H} \le 1 \times 10^{-7} \qquad \text{(uniformly distributed)} \tag{3.42}$$

$$0 \le \frac{\delta x}{x} \le 1 \times 10^{-7} \qquad \text{(uniformly distributed)}$$

In Appendix A it is shown that the improvement in the error propagation is in the ratio of

$$\Delta H\left(\frac{e^{\Delta H}}{e^{\Delta H} - 1}\right) \tag{3.43}$$

For a transfer from the sphere of influence to a peripoint near a planet ΔH can be of the order of 10. The ratio (3.43) shows that the expected improvement in accuracy is the same order of magnitude.

The new formulation Equations (3.30) through (3.39) are used in the program for the case Δt (or ΔH) > 0. In the case $\Delta t < 0$ the variable x defined in Eq. (3.29) ranges only in the interval (1,0) for all negative Δt (or ΔH), and precision would be lost in using that form. An obvious alternative is to define

$$1 + x = e^{-\Delta H} \text{ for } \Delta t < 0$$

and rederive the above equations. However, it turns out to be simpler and just as accurate to reverse the sign of $\underline{v}_0$ and Δt, compute a negative $\underline{v}(t)$ using the above equations, and then reverse its sign to obtain the final desired value. This is the method used in the program.

3.5 Time Step Computation Method

The time step is computed in the program from the formula

$$h_{j+1} = K \left[\text{tr}(G_j G_j^T) \right]^{-\frac{1}{4}} \tag{3.44}$$

where $h_{j+1} = t_{j+1} - t_j$ is the time step between the jth and (j+1)th points

G_j is the gravity gradient matrix at the jth point

K is the scaling constant set by an input data card (a typical value is about 0.1 in the single-precision version).

This formula is derived by an argument based on the state transition matrix integration. This computation is very sensitive to numerical inaccuracies, and it turns out that a time step which is chosen for sufficient accuracy in the state transition matrix integration will also give satisfactory accuracy in the state integration by the Encke method. The method of derivation is explained in Appendix B. There is a further useful relation for computing the trace in (3.44) which should be pointed out here. Let g_{mn} be the (m,n)th element of G

$$G = \left[g_{mn} \right]$$

Then the trace is computed by

$$\text{tr}(GG^T) = \sum_{m=1}^{3} \sum_{n=1}^{3} g_{mn}^2 \tag{3.45}$$

Since G varies approximately as $1/r^3$, where r is the distance of the spacecraft from the primary body, the time step varies approximately as $r^{3/2}$.

In planet-centered hyperbolic phases two special problems must

be considered. Firstly, it is found that the magnitudes of the G matrix elements vary considerably over a time step, so that an"average" value of G must be used for each time step in order to make the time step lengths consistent inbound and outbound. This is done in the program by a linear differential correction to the value given by (3.44). Secondly, the computed time steps can be very large in the region near the asymptotes of a hyperbolic trajectory. Since the phase test is made in the program at the end of each time step, the phase change from planet to sun-centered coordinates can be unnecessarily delayed unless a limit is placed on the allowable time step magnitude. A limit of one day in hyperbolic phases is used in the program.

In the double-precision version of the program the time step is computed in the same manner, except that the scale factor K is reduced. A typical value of K is about .01 in the double-precision version.

3.6 Trajectory Phase Determination Method

The trajectory phase designates which of the solar system bodies is the primary body. The phase determination is based on the gravitational gradient matrix G defined in (3.10) through (3.12). The starting phase in a trajectory is always known from the specification of the starting planet (or the Sun) in the input data. Thereafter, a phase test is made at the end of each time step in the integration. The test consists of comparing $tr(G_{pri}G^T_{pri})$ with $tr(G_{pert}G^T_{pert})$, where G_{pri} and G_{pert} are the two components of G defined in (3.11) and (3.12). As long as

$$tr(G_{pri}G^T_{pri}) > tr(G_{pert}G^T_{pert})$$

the current phase remains unchanged. When the relation above does not hold, a phase change occurs.

The phase change procedure is as follows. If the old primary body were a planet, the new primary body is the Sun. If, on the other hand, the old primary body were the Sun, then the nearest planet to the spacecraft is selected to be the new primary body. The planets are never near enough together so that an ambiguity can occur in this selection procedure. The coordinate system origin is changed from the old to the new primary body, and this, of course, requires appropriate combination of spacecraft and primary body position and velocity vectors. However, since no coordinate rotation is done, no change in the state transition matrix occurs at phase change points.

3.7 Trajectory Search Method

The program will perform an iterative search for a trajectory between a starting point and an endpoint with a fixed time of flight. The search works for either forward or backward integration. The submatrix N of the state transition matrix in Equation (3.7) relates endpoint position deviations to beginning point velocity variations, and the N matrix is used to implement the search procedure.

The first search iteration begins at the fixed starting point

with an initial guess for the starting velocity, and a trajectory is computed toward the target point. The trajectory terminates when the fixed time of flight is reached, and the position miss vector $\Delta\underline{r}_m$ is computed:

$$\Delta\underline{r}_m = \underline{r}(t_f) - \underline{r}_{targ} \tag{3.46}$$

where $\underline{r}(t_f)$ is the computed trajectory endpoint at the time of flight t_f.

$\underline{r}_{targ}$ is the intended target point.

Equation (3.7), where i denotes the starting point and j is now the endpoint f, gives a linear approximation to the deviation in initial velocity which causes the miss $\Delta\underline{r}_m$:

$$\Delta\underline{v}_i = N_{fi}^{-1}\Delta\underline{r}_m \tag{3.47}$$

Therefore, in order to null the miss, a new starting velocity is computed for the next trajectory iteration by simply subtracting $\Delta\underline{v}_i$ from the initial starting velocity. This same procedure continues for subsequent iterations, and the general relation for the starting velocity for the (n + 1)th iteration in terms of the miss computed on the nth iteration is

$$\underline{v}_{i_{n+1}} = \underline{v}_{i_n} - N_{fi_n}^{-1}\Delta\underline{r}_{m_n} \tag{3.48}$$

Note that since a new state transition matrix is computed in each iteration, the correct N_{fi} is available at the end of each iteration automatically. The search terminates when the magnitude of the miss vector becomes less than (or equal to) a specifiable tolerance which

is set by an input data card.

The search computations are linear, yet the search is relatively efficient because the state transition matrix is updated in each iteration. The linear search works quite well when the miss is so large that the endpoint is outside the sphere of influence of the target planet. In that case the target point becomes referenced to the Sun, rather than the planet, and the non-linear planet influences are not strong. However, inside the sphere of influence of the target planet the problem becomes highly non-linear if the target point is near the planet because the miss vector $\Delta\underline{r}_m$ may be as large as, or even larger than, the endpoint position vector in the planet-referenced coordinates.

In most cases tested thus far even in the most non-linear region the linear search computations reduce the miss by at least a factor of two with each iteration. These tests also indicate good convergence when the initial guess trajectory misses the target planet by as much as one-half the radial distance of the planet from the Sun. Table 3-1 shows some typical results of trajectory searches which have been performed.

Nonconvergence of the search is experienced often in orbits in which the heliocentric transfer angle is near 180 degrees. The reason is the singularity in the computation of the out-of-plane velocity correction; the linear search computations break down as the transfer angle nears 180 degrees unless the first guess for the out-of-plane initial velocity component is very good. When such a case occurs, it is necessary to vary the out-of-plane initial velocity component manually until a value is found for which the search converges. The 258 day Mars transfer listed in Table 3-1 has a heliocentric transfer angle of approximately 178 degrees, and illustrates that the search can be done successfully by this method.

Since the search converges well for large initial misses, the determination of an initial guess for a starting velocity is not highly critical. However, the better the initial guess, the more efficient the search will be, clearly. The following simple patched-conic procedure has been successfully used to obtain an initial starting velocity:

1. Find a heliocentric elliptical transfer between the target position on the arrival date and the initial position on the starting date. If these two positions are near planets, use the planet positions for the first guess.

Table 3-1 Trajectory Search Results Summary (Single-Precision Program)

Trajectory	Initial Miss	No. of Iterations to Reach Noise Level	RMS Position Noise Level
258 day Earth to Mars target point 5000 km above Mars surface.	6×10^5 km	7	65km
4 day Mars sphere of influence to Mars. Target point 5000 km above Mars surface. (Initial conditions from 258 day transfer above.)	9.5	2	0.2
120 day Earth to Venus transfer. Target point 7500 km above Venus surface	1.9×10^7	6	23

2. Compute the elliptical velocity at the initial point. If the initial point is outside the sphere of influence of a planet, this is the initial guess velocity. Otherwise a further computation is necessary.

3. Refer this elliptical velocity to the starting planet by simply subtracting the planet's velocity from the elliptical velocity vectorially. It is well to use the planet's velocity a few days after injection to account for the time to reach the sphere of influence from injection. This effects an angular correction of a few degrees which considerably refines the initial guess. The resulting velocity referenced to the starting planet is taken to be the asymptotic velocity for a hyperbolic transfer from the injection point to the sphere of influence.

4. Compute the injection velocity at the injection point (which should be previously chosen or specified) to achieve the asymptotic velocity computed in step 3. This result is the initial guess velocity.

Some caution must be used in this procedure to ensure that the target point is accessible (if it is near a planet) by a hyperbolic approach from the direction of the elliptical transfer. Also, the injection point with respect to the launch planet must be chosen to make the asymptotic velocity achievable by a hyperbolic transfer. Reference 3 gives all pertinent conic equations for these computations.

CHAPTER IV

PROGRAM STRUCTURE AND FUNCTIONAL FLOW

The explanation of the program structure and functional flow given in this chapter relates directly to the single-precision version. The functional organization of the double-precision version is identical with that of the single-precision version, but the detailed program structures are different in that the double-precision version requires some additional support subprograms.

The single-precision version is organized into a main control program, designated as MAIN, and the following subprograms:

INTEG ENCØN GRAVFØ EPHEM ØSCØN PRINTT TIME AXB	Subroutine subprograms
VMAG ADØTB	Function subprograms

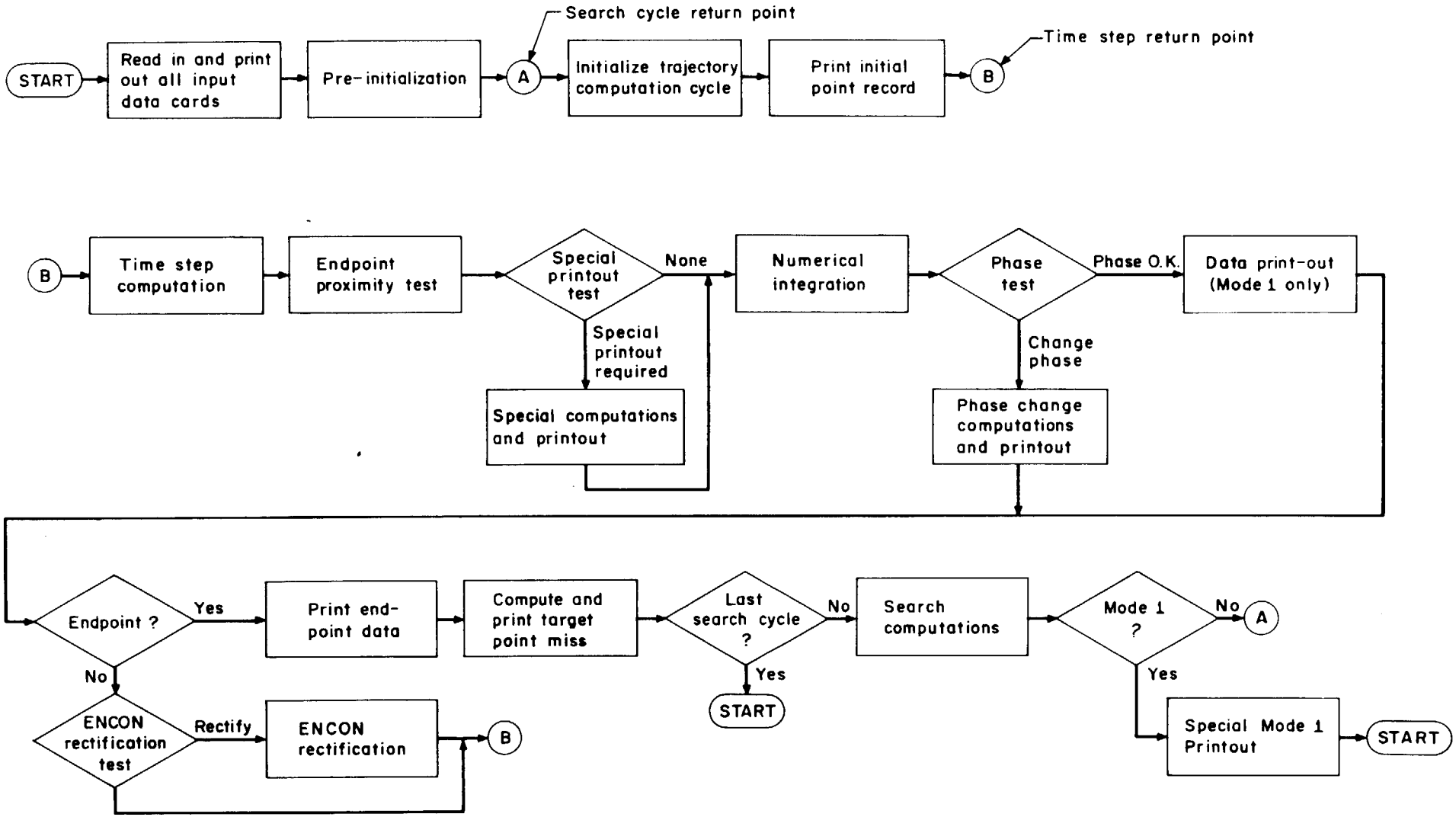
This chapter describes the primary functions of MAIN and each of the subprograms and shows the program functional flow.

4.1 MAIN Program

The MAIN program establishes and controls the entire functional flow. Figure 4-1 is the program functional flow diagram mechanized by MAIN. The first operation is reading in and printing out the information on the input data cards. This makes available all input data in original format for convenient reference.

All initialization operations on the input data which are unchanged in a trajectory search are performed in the pre-initialization functional block. These include conversion of times entered in days,

Figure 4-1 Program Functional Flow



hours, minutes, and seconds into seconds referenced to start time, setting up invariant constants used in the program, initializing the ephemeris computation subroutine (EPHEM), and printing an identification record for the trajectory computation to be performed.

The initializing operations which are repeated at the beginning of each search cycle include setting up phase-dependent constants, initial conditions for the state and state transition matrix integrations, and initialization of the Encke conic computation subroutine (ENCØN). These are performed following the search cycle return point designated as point A in Figure 4-1.

An initial point record is printed following the initialization for each of the following three conditions: (1) Mode 1 is the operating mode; (2) the last search cycle in Modes 2 or 3; (3) each search cycle if an input data control constant has been set to cause a printout each cycle of the search. This record has the standard information described in Section 2.1.6, and it lists all trajectory initial conditions.

Point B in Figure 4-1 is the return point on an inner loop in the program which is recycled with each time step. The first operation in this loop is the time step computation by the method described in Section 3.5. This is followed by an endpoint proximity test to determine if the trajectory end time is less than twice the time step from the present time. If it is, the time step is modified to be half the remaining time and appropriate flags are set up to effect termination on the following cycle. Otherwise, the next functional test is entered immediately.

The special printout test determines whether a time point for a special printout lies within the present time step. If so, the special computations and printout occur as described in Section 2.1.6.4.

The numerical integration is performed in the INTEG subroutine which in turn calls ENCØN to compute the conic position and velocity used in the Encke method, EPHEM to compute the planet positions and GRAVFØ to compute the gravitational perturbing acceleration and gravity gradient matrix used in the integration.

A phase test occurs at the end of each integration step as described in Section 3.6 to determine whether the current primary body is correct. If not, a phase change occurs, involving the determination of the new reference body, computation of velocity and position of the reference body, computation of velocity and position of the reference bodies by the EPHEM subroutine, and printout of old and new phase in-

formation as described in Section 2.1.6. In Mode 1 the normal time step data printout follows the phase test immediately, if no phase change occurs. The trajectory endpoint is indicated by a flag set earlier in the endpoint proximity test. If the endpoint has not been reached, the ENCØN rectification test is performed, ENCØN is rectified if necessary, and a transfer takes place to begin a new time step. The rectification test is simply described with reference to equation (3.3); rectification must occur if

$$|\underline{\delta}| > \frac{|\underline{r}|}{100} \quad \text{(single-precision version)}$$

Rectification is simply reinitialization of the Encke conic at the current true state.

If the endpoint has been reached, an endpoint data printout occurs if the operating mode is Mode 1, or if this endpoint ends the last search cycle in Modes 2 or 3, or if there is a command to print every search cycle. This printout is followed by computation and printout of the target point miss.

If the current cycle is the last search cycle, the trajectory computations terminate for the present case by transferring program control to START to enable a subsequent case to begin. Otherwise, the search computations are performed, as described in section 3.7, to update the starting velocity. Then, if the operating mode is Mode 1, the search computation results are printed as described in Section 2.1.6 and control is transferred to START. Otherwise, the next cycle of the trajectory search begins by returning to the initialization of the new cycle.

4.2 INTEG Subroutine

INTEG performs the numerical integration of both the state and the state transition matrices by the methods described in Sections 3.1 through 3.3.

INTEG is called by the MAIN program in both the numerical integration functional block and special data printout loop shown in Figure 4-1. The call statement is

CALL INTEG (H)

The argument H is the time step computed in MAIN. INTEG returns the position and velocity increments $\underline{\delta}$ and $\underline{\nu}$ defined in Section 3.1 and the A and B matrices defined in Section 3.3. This information, as well as other information shared between two or more subprograms, is

placed in CØMMØN storage.

INTEG in turn calls ENCØN, EPHEM, and GRAVFØ.

4.3 ENCØN Subroutine

ENCØN computes the conic position and velocity used in the Encke integration method. There are two primary branches in the subroutine, one for rectification and one for computation of the conic state. The computation branch has two sub-branches, one for ellipses and one for hyperbolae. The call statement is

CALL ENCØN (IENC)

The argument IENC is a control constant which selects the desired branch. If IENC is 1, the rectification branch is selected. Rectification makes use of the current true state to determine a new osculating (Encke) conic to be used in all subsequent computations until rectification reoccurs. If IENC is 0, the computation branch is selected. If the Encke conic is an ellipse, the state computations are performed by a normal method (Chapter 2, reference 3); if the conic is an hyperbola, the special formulation described in Section 3.4 is used.

All variables used and the conic state returned by ENCØN are placed in CØMMØN storage. ENCØN is called by MAIN for purposes of rectification and by INTEG for conic state computation. ENCØN in turn calls TIME and GRAVFØ in the rectification branch and prints a rectification message giving the time of rectification and identifying the Encke conic type.

4.4 GRAVFØ Subroutine

This subroutine computes the right hand side of Equation (3.4) used in the state integration and the gravitational gradient matrix G_{pri} and G_{pert} defined in equations (3.10) through (3.12) used in the state transition matrix integration. GRAVFØ is called normally by INTEG for these computations, and it is also called by ENCØN at each rectification point to reinitialize the computations. The call statement is

CALL GRAVFØ (UA,GFØRCE)

UA is a three-dimensional input vector corresponding to $\underline{\delta}$ in equation (3.4). GFØRCE is a three-dimensional vector which is the acceleration (right hand side of equation (3.4)) returned by GRAVFØ. All other quantities used and returned by the subroutine are in CØMMØN storage. GRAVFØ calls no other subroutines.

4.5 EPHEM Subroutine

This subroutine computes the planet ephemerides by the osculating conic method described in Section 2.1.4. There are two branches, one for initialization and one for computation. The initialization branch establishes the osculating ellipses approximating the planet orbits from the reference ephemerides entered as program input data.

EPHEM is called by MAIN for initialization and for computation of primary body ephemerides at the phase change points, and by INTEG for computation of planet positions for disturbing force computation. The call statement is

CALL EPHEM (N, IEPH, UP, VP)

N is the identifying number of the planet whose ephemerides are to be computed. IEPH is a constant controlling the following operations according to the specified value:

2 - initialization

1 - computation of planet position and velocity (required at phase change points only)

0 - computation of planet position only (position only is required in perturbing acceleration and gravity gradient matrix computations).

UP and VP are, respectively, three-dimensional position and velocity returned by the subroutine.

EPHEM calls no other subroutines.

4.6 PRINTT Subroutine

This subroutine does all of the normal printing for the program described in Section 2.1.6. PRINTT is called by MAIN only. The call statement is

CALL PRINTT(IPRINT,NØSC)

IPRINT is a control constant selecting the record format to be printed. NØSC is a control constant which determines whether ØSCØN is called following the printout. PRINTT also calls the TIME subroutine. Further detailed description is left to Reference 1.

4.7 ØSCØN Subroutine

ØSCØN computes the osculating conic data described in Section 2.1.7. ØSCØN is an optional subroutine; a control constant in the program input data determines whether or not the subroutine is used. ØSCØN is called only by the PRINTT subroutine, and, if it is to be

used in the program, it is called each time a data printout occurs. Further detailed description of the computations is left to Reference 1.

4.8 TIME Subroutine

This subroutine converts time, which is in units of seconds in the program computations, into days, hours, minutes, and seconds. It is used always for the purpose of printing time in the more useful form. TIME is called by ENCØN and PRINTT. The call statement is

CALL TIME(TARG,JDAY,JHR,JMIN,XSEC)

TARG is the input time argument in seconds. The other arguments in the call statement are, respectively, the days, hours, minutes, and seconds returned by the subroutine. The input time argument is negative for trajectory integrations backward in time; however, the days, hours, minutes, and seconds returned are always positive, and they must be interpreted as the elapsed time from start in either the positive or negative direction as the case may be.

4.9 AXB Subroutine

AXB computes the cross product of two vectors. It is called by MAIN and several of the other subroutines where vector products are required in the computation. The call statement is

CALL AXB(A,B,VPRØD)

A and B are three-dimensional vectors, and VPRØD is the three-dimensional vector product returned by the subroutine. The vector computation is

$$\underline{VPRØD} = \underline{A} \times \underline{B}$$

4.10 VMAG Function Subprogram

VMAG computes the magnitude of a vector. It is used throughout the program wherever vector magnitudes are required in computations. The specification statement is

VMAG(V)

V is a three-dimensional vector, and the subprogram returns the square root of the sum of the squares of the three components.

4.11 ADØTB Function Subprogram

ADØTB computes the scalar product of two vectors. It is used throughout the program wherever scalar products are required in computations. The specification statement is

ADØTB(A,B)

A and B are three-dimensional vectors, and the program returns the scalar product $\underline{A} \cdot \underline{B}$.

APPENDIX A

DERIVATION OF NEW FORMULATION OF KEPLER'S PROBLEM FOR HYPERBOLAE

The purpose of this Appendix is to derive the equations of the new formulation in the form given in Section 3.4 and to show the improvement in the error propagation characteristics. The equations of the standard conic formulation of Kepler's problem for hyperbolae given in Section 3.4 are repeated here for convenient reference:

Kepler's equation:

$$\sqrt{\frac{\mu}{a^3}}\,\Delta t = -\Delta H + \frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}}\left(\cosh \Delta H - 1\right) + \left(1 + \frac{r_0}{a}\right)\sinh \Delta H \tag{3.23}$$

State equations:

$$\underline{r}_{enc}(\Delta H) = \left[1 - \frac{a}{r_0}\left(\cosh \Delta H - 1\right)\right]\underline{r}_0 + \left[\frac{a}{\mu}\,\underline{r}_0 \cdot \underline{v}_0\left(\cosh \Delta H - 1\right) + r_0\sqrt{\frac{a}{\mu}}\sinh \Delta H\right]\underline{v}_0 \tag{3.24}$$

$$\underline{v}_{enc}(\Delta H) = -\frac{\sqrt{\mu a}}{r r_0}\sinh \Delta H\,\underline{r}_0 + \left[1 - \frac{a}{r}\left(\cosh \Delta H - 1\right)\right]\underline{v}_0 \tag{3.25}$$

(In this derivation equations appearing in Section 3.4 will have Section 3 equation numbers.)

The derivation begins with Kepler's equation. Let

$$t_m = \sqrt{\frac{\mu}{a^3}}\,\Delta t \tag{3.35}$$

Expressing $\cosh \Delta H$ and $\sinh \Delta H$ in terms of $e^{\Delta H}$ and $e^{-\Delta H}$ and gathering like terms gives:

$$t_m = -\Delta H - \frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}} + \frac{1}{2}\left[\frac{\underline{r}_0 \underline{v}_0}{\sqrt{\mu a}} + \left(1 + \frac{r_0}{a}\right)\right]e^{\Delta H} + \frac{1}{2}\left[\frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}} - \left(1 + \frac{r_0}{a}\right)\right]e^{-\Delta H} \tag{A.1}$$

Define K_1 and K_3, constants for a given orbit:

$$K_1 = \frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}} + \left(1 + \frac{r_0}{a}\right) = \frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}} + \left(\frac{r_0 v_0^2}{\mu} - 1\right) \tag{3.32}$$

$$K_3 = \frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}} - \left(1 + \frac{r_0}{a}\right) = \frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}} - \left(\frac{r_0 v_0^2}{\mu} - 1\right) \tag{A.2}$$

The equivalence expressed in these equations is readily established from the well known energy integral for hyperbolae:

$$v_0^2 = \mu\left(\frac{2}{r_0} + \frac{1}{a}\right)$$

The following properties of K_1 and K_3 are also readily verified from well known conic relations:

(1) $K_1 K_3 = -e^2$

Where e is the eccentricity of the hyperbola

$$e^2 = 1 + \frac{h^2}{\mu a} = \left(1 + \frac{r_0}{a}\right)^2 - \left(\frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}}\right)^2$$

$h = \left|\underline{r}_0 \times \underline{v}_0\right|$ angular momentum per unit mass in orbit

(2) $K_1 > 0$ always, since $1 + \frac{r_0}{a} > \frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}}$ for $e^2 > 0$

$K_3 < 0$ always for the same reason.

Make the following change of variable in (A. 1):

$$e^{\Delta H} = 1 + x, \quad \Delta H = \log(1 + x) \tag{3.29}$$

and use (3.32) and (A. 2) for the coefficients:

$$t_m = -\log(1+x) - \frac{K_1 + K_3}{2} + \frac{K_1}{2}(1 + x) + \frac{K_3}{2}\left(\frac{1}{1+x}\right) \tag{A.3}$$

Multiplying by 1 + x and clearing terms

$$(1+x)\,t_m = -(1+x)\log(1+x) - \frac{K_3}{2}x + \frac{K_1}{2}x(1+x) \qquad (A.4)$$

Regrouping the terms:

$$\frac{K_1}{2}x^2 + \left(\frac{K_1 - K_3}{2} - t_m\right)x - t_m - (1+x)\log(1+x) = 0 \qquad (A.5)$$

Referring to (3.32) and (A.2), define K_2 by

$$K_2 = \frac{K_1 - K_3}{2} - t_m = \left(1 + \frac{r_0}{a}\right) - t_m = \frac{r_0 v_0^2}{\mu} - 1 - t_m$$

Then Kepler's equation becomes

$$\frac{K_1}{2}x^2 + K_2 x - (1+x)\log(1+x) = t_m \qquad (3.30)$$

K_1 can be computed with high precision by the appropriate choice of two different methods. If $\underline{r}_0 \cdot \underline{v}_0 > 0$, K_1 is computed by Equation (3.32) directly, If $\underline{r}_0 \cdot v_0 < 0$, then K_1 can be a small difference of the two terms in (3.32), and a more precise computation is

$$K_1 = -\frac{1 + \frac{|\underline{r}_0 \times \underline{v}_0|^2}{\mu a}}{\frac{\underline{r}_0 \cdot \underline{v}_0}{\sqrt{\mu a}} - \left(1 + \frac{r_0}{a}\right)} \qquad (\underline{r}_0 \cdot \underline{v}_0 < 0) \qquad (3.33)$$

The denominator of (3.33) is large and negative for this case.

K_2 in Equation (3.30) can normally be computed precisely in regions where (3.30) would be sensitive to errors in K_2. K_2 becomes imprecise when $t_m \approx \left(1 + \frac{r_0}{a}\right)$, but then the $\frac{K_1}{2}x^2$ term dominates, greatly reducing the sensitivity to error in K_2.

There is, however, a case in which (3.30) is imprecise for small x. Consider an outbound transfer with initial point near the peripoint. Then r_0/a is of the order of 1, x will be small for the first time step, and t_m is very small. Then K_2 is of the order of 2, and t_m is a function of a small difference between the

second and third term on the left of (3.3) and the second-order first term. Consequently, for the case of small x (3.30) is reformulated so that the left side is separated into first, second, and higher order terms explicitly. We begin by putting it into the form

$$\left(\frac{K_1 - 1}{2}\right) x^2 + (K_2 - 1)\, x + \frac{x^2}{2} + x - (1 + x) \log (1 + x) = t_m \qquad \text{(A. 6)}$$

The last two terms in (A. 6) can be combined into the following power series expansion:

$$\frac{x^2}{2} + 2 - (1 + x) \log (1 + x) = x^3 \sum_{k=0}^{\infty} \frac{x^k (-1)^k}{(k+2)(k+3)}$$

$$= x^3 L(x) \qquad (0 \leq x < 1) \qquad \text{(A. 7)}$$

This relation is easily verified by expanding log (1+x) in a power series and combining like terms on the left side. (A. 7) is positive and well behaved at x = 0. Kepler's equation is finally expressed as

$$(K_2 - 1)\, x + \left(\frac{K_1 - 1}{2}\right) x^2 + x^3 L(x) = t_m \quad (x < 0.3) \qquad (3.31)$$

The range limit is established simply by noting that (3.30) is sufficiently precise for larger values of x.

The state Equations (3.24) and (3.25) are expressed as functions of x directly by substitution of the following identities:

$$\cosh \Delta H - 1 = \frac{x^2}{2(1 + x)} \qquad \text{(A. 8)}$$

$$\sinh \Delta H = \frac{2x + x^2}{2(1 + x)} \qquad \text{(A. 9)}$$

The results are

$$\underline{r}_{enc}(x) = \left[1 - \frac{a}{2r_0}\left(\frac{x^2}{1 + x}\right)\right] \underline{r}_0 + \left[\frac{a}{2\mu}\, \underline{r}_0 \cdot \underline{v}_0 \left(\frac{x^2}{1 + x}\right) + r_0 \sqrt{\frac{a}{\mu}}\, \frac{x\left(1 + \frac{x}{2}\right)}{1 + x}\right] \underline{v}_0 \qquad (3.38)$$

$$\underline{v}_{enc}(x) = -\frac{\sqrt{\mu a}}{r r_0}\, \frac{x\left(1 + \frac{x}{2}\right)}{1 + x}\, \underline{r}_0 + \left[1 - \frac{a}{2r}\left(\frac{x^2}{1 + x}\right)\right] \underline{v}_0 \qquad (3.39)$$

The propagation characteristics of errors in the iteration variables in the two formulations are developed for a generalized comparison in the following manner. Consider first an error in $\underline{r}_{enc}(\Delta H)$, $\delta \underline{r}_{enc}(\Delta H)$, caused by an error in the iteration variable ΔH in Equation (3.24) compared with an error in $\underline{r}_{enc}(x)$, $\delta \underline{r}_{enc}(x)$, caused by an error in the iteration variable x. That is, for a given transfer time Δt we wish to compare the errors in the computed $\underline{r}_{enc}(\Delta t)$ obtained from the two formulations. Eqs. (3.27) and (3.40) give $\delta \underline{r}_{enc}(\Delta H)$ and $\delta \underline{r}_{enc}(x)$. They are rewritten here in the following forms:

$$\delta \underline{r}_{enc}(\Delta H) = f_1(\Delta H)\, \delta(\Delta H)\, \underline{r}_0 + f_2(\Delta H)\, \delta(\Delta H)\, \underline{v}_0 \tag{A.10}$$

$$\delta \underline{r}_{enc}(x) = g_1(x)\, \delta x\, \underline{r}_0 + g_2(x)\, \delta x\, \underline{v}_0 \tag{A.11}$$

where

$$\begin{aligned} f_1(\Delta H) &= -\frac{a}{r_0} \sinh \Delta H \\ f_2(\Delta H) &= \frac{a}{\mu}\, \underline{r}_0 \cdot \underline{v}_0 \sinh \Delta H + r_0 \sqrt{\frac{a}{\mu}} \cosh \Delta H \\ g_1(x) &= -\frac{a}{2r_o} \frac{2x + x^2}{(1+x)^2} \\ g_2(x) &= \frac{a}{2\mu}\, \underline{r}_0 \cdot \underline{v}_0 \frac{2x+x^2}{(1+x)^2} + r_0 \sqrt{\frac{a}{\mu}} \frac{1+x+\frac{x^2}{2}}{(1+x)^2} \end{aligned} \tag{A.12}$$

The covariance matrices for the errors can then be written as follows:

$$\begin{aligned} E_{\Delta H} &= \overline{\left[\delta \underline{r}_{enc}(\Delta H) - \overline{\delta \underline{r}_{enc}(\Delta H)}\right]\left[\delta \underline{r}_{enc}(\Delta H) - \overline{\delta \underline{r}_{enc}(\Delta H)}\right]^T} \\ &= \sigma^2_{\Delta H} \left\{ f_1^2\, \underline{r}_0 \underline{r}_0^T + f_1 f_2 \left(\underline{r}_0 \underline{v}_0^T + \underline{v}_0 \underline{r}_0^T \right) + f_2^2\, \underline{v}_0 \underline{v}_0^T \right\} \end{aligned} \tag{A.13}$$

$$\begin{aligned} E_x &= \overline{\left[\delta \underline{r}_{enc}(x) - \overline{\delta \underline{r}_{enc}(x)}\right]\left[\delta \underline{r}_{enc}(x) - \overline{\delta \underline{r}_{enc}(x)}\right]^T} \\ &= \sigma_x^2 \left\{ g_1^2\, \underline{r}_0 \underline{r}_0^T + g_1 g_2 \left(\underline{r}_0 \underline{v}_0^T + \underline{v}_0 \underline{r}_0^T \right) + g_2^2\, \underline{v}_0 \underline{v}_0^T \right\} \end{aligned} \tag{A.14}$$

These are matrix equations and the terms within the brackets are all constants. It will now be shown that $E_{\Delta H}$ is related to E_x by a scalar constant.

As was discussed in Section 3.4, the iteration variable errors $\delta(\Delta H)$ and δx are uniformly distributed within the iteration tolerance:

$$0 \leq \frac{\delta(\Delta H)}{\Delta H} \leq 1 \times 10^{-7}$$

$$0 \leq \frac{\delta x}{x} \leq 1 \times 10^{-7} \tag{3.42}$$

From these relations we can write

$$\delta(\Delta H) = \Delta H \, \delta n \tag{A. 15}$$

$$\delta x = x \, \delta n \tag{A. 16}$$

where δn is a random variable uniformly distributed over the interval$(0, 1 \times 10^{-7})$. Then

$$\sigma^2_{\Delta H} = \overline{\delta(\Delta H)^2} - \overline{\delta(\Delta H)}^2 = (\Delta H)^2 \left[\overline{\delta n^2} - \overline{\delta n}^2\right] = (\Delta H)^2 \sigma_n^2 \tag{A. 17}$$

$$\sigma_x^2 = \overline{\delta x^2} - \overline{\delta x}^2 = x^2 \sigma_n^2 \tag{A. 18}$$

Substituting (A. 17) and (A. 18) into (A. 13) and (A. 14):

$$E_{\Delta H} = (\Delta H)^2 \sigma_n^2 \left\{ f_1^2 \underline{r}_0 \underline{r}_0^T + f_1 f_2 \left(\underline{r}_0 \underline{v}_0^T + \underline{v}_0 \underline{r}_0^T \right) + f_2^2 \underline{v}_0 \underline{v}_0^T \right\} \tag{A. 19}$$

$$E_x = x^2 \sigma_n^2 \left\{ g_1^2 \underline{r}_0 \underline{r}_0^T + g_1 g_2 \left(\underline{r}_0 \underline{v}_0^T + \underline{v}_0 \underline{r}_0^T \right) + g_2^2 \underline{v}_0 \underline{v}_0^T \right\} \tag{A. 20}$$

The right sides of Equations (A. 19) and (A. 20) consist of the sum of three matrix terms, each with a scalar coefficient. We now consider the ratios of corresponding coefficients. The ratio of the coefficients of the first terms is (substituting from (A. 12), (A. 9) and (3. 29):

$$\frac{(\Delta H)^2 f_1^2}{x^2 g_1^2} = \frac{(\Delta H)^2 \sinh^2 \Delta H}{x^2 \dfrac{(2x + x^2)^2}{4(1+x)^4}}$$

$$= \frac{(\Delta H^2 \dfrac{(2x + x^2)^2}{4(1+x)^2}}{x^2 \dfrac{(2x + x^2)^2}{4(1+x)^4}} = \frac{(\Delta H)^2 (1+x)^2}{x^2}$$

$$= (\Delta H)^2 \left(\frac{e^{\Delta H}}{e^{\Delta H} - 1} \right)^2 \tag{A. 21}$$

The ratio of coefficients of the second terms is:

$$\frac{(\Delta H)^2 f_1 f_2}{x^2 g_1 g_2} = \frac{(\Delta H)^2 \sinh \Delta H \left[\frac{a}{\mu}\, \underline{r}_0 \cdot \underline{v}_0 \sinh \Delta H + r_0 \sqrt{\frac{a}{\mu}} \cosh \Delta H\right]}{x^2 \left(\frac{2x+x^2}{2(1+x)^2}\right)\left[\frac{a}{2\mu}\, \underline{r}_0 \cdot \underline{v}_0 \frac{2x+x^2}{(1+x)^2} + r_0 \sqrt{\frac{a}{\mu}} \frac{1+x+\frac{x^2}{2}}{(1+x)^2}\right]}$$

$$= \frac{(\Delta H)^2 (1+x)^2}{x^2}$$

$$= (\Delta H)^2 \left(\frac{e^{\Delta H}}{e^{\Delta H}-1}\right)^2 \qquad \text{(A. 22)}$$

The ratio of coefficients of the third terms is:

$$\frac{(\Delta H)^2 f_2^2}{x^2 g_2^2} = \frac{(\Delta H)^2 \left[\frac{a}{\mu}\, \underline{r}_0 \cdot \underline{v}_0 \sinh \Delta H + r_0 \sqrt{\frac{a}{\mu}} \cosh \Delta H\right]^2}{x^2 \left[\frac{a}{2\mu}\, \underline{r}_0 \cdot \underline{v}_0 \frac{2x+x^2}{(1+x)^2} + r_0 \sqrt{\frac{a}{\mu}} \frac{1+x+\frac{x^2}{2}}{(1+x)^2}\right]^2}$$

$$= \frac{(\Delta H)^2 (1+x)^2}{x^2} = (\Delta H)^2 \left(\frac{e^{\Delta H}}{e^{\Delta H}-1}\right)^2 \qquad \text{(A. 23)}$$

It is evident from (A. 21) through (A. 23) that $E_{\Delta H}$ and E_x are related by a scalar multiplier:

$$E_{\Delta H} = \left(\frac{\Delta H\, e^{\Delta H}}{e^{\Delta H}-1}\right)^2 E_x = k^2 E_x \qquad \text{(A. 24)}$$

The limiting behavior of k is

1) $\lim_{\Delta H \to 0} k = 1$

2) For ΔH large, $k \to \Delta H$

Consequently, the effects of iteration errors in the new x formulation are always less than the effects of iteration errors in the ΔH formulation for all transfers $\Delta H > 0$.

The same analysis can be carried through for $\delta \underline{v}_{enc}(\Delta H)$ and $\delta \underline{v}_{enc}(x)$, with the same constant ratio resulting between the corresponding covariance matrices.

APPENDIX B

TIME STEP FORMULA DERIVATION

This Appendix describes the derivation of the time step Equation (3.44), repeated here for convenient reference:

$$h_{j+1} = K \left[tr\,(G_j G_j^T) \right]^{-\frac{1}{4}} \tag{3.44}$$

where $h_{j+1} = t_{j+1} - t_j$ is the time step between the jth and (j+1)th points

G_j is the gravity gradient matrix at the jth point

K is a constant

The method of derivation begins with the assumption that G_j is constant over a local region of the trajectory about the jth point. This permits a simple analytical determination of the state transition matrix between the jth and (j+1)th points. The error in each term of the numerically integrated state transition matrix is then taken to be approximately equal to the 5th order term in the Taylor series expansion of the analytical result, since the Nyström technique is accurate to fourth order. Requiring that this error be less than 10^{-8} times the analytic result yields an expression from which Equation (3.44) can be generalized.

The assumption that the gravity gradient matrix is constant over local regions is relatively good in the midcourse phase of a trajectory. In the phases of a trajectory near a planet G does change significantly within a time step. In this case, Equation (3.44) is still used, but a simple first order differential correction to h is introduced. The derivation suggests a value for the constant K; however, K has been made a program control constant so that time step scaling can be done by means of an input data card.

The derivation outlined above begins with Equation (3.9)

$$\delta \ddot{\underline{r}}_j = G_j \, \delta \underline{r}_j \tag{3.9}$$

The matrix G_j defined in Equations (3.10) through (3.12) is real and symmetric. It therefore has real eigenvalues, which may be positive, negative, or zero, and eigenvectors which can be orthonormalized. Then, a diagonalizing matrix Q

can always be found to transform (3.9) into

$$\delta \ddot{\underline{z}}_j = Q^T \delta \ddot{\underline{r}}_j = Q^T G_j Q \ \delta \underline{z}_j \tag{B.1}$$

where $Q^T G_j Q$ is a diagonal matrix with the eigenvalues of G_j along the principle diagonal.

$$Q^T G_j Q = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \tag{B.2}$$

In this form the three component equations of (B.1) can be considered separately. Consider, for example, the first component equation:

$$\delta \ddot{z}_1 = \lambda_1 \delta z_1 \tag{B.3}$$

The following solutions are possible:

$\underline{\lambda_1 < 0}$:

$$\begin{aligned} \delta z_1 (t_j + h) &= \delta z_1 (t_j) \cos\sqrt{\lambda_1}\ h + \frac{\delta \dot{z}_1 (t_j)}{\sqrt{\lambda_1}} \sin\sqrt{\lambda_1}\ h \\ \delta \dot{z}_1 (t_j + h) &= \delta \dot{z}_1 (t_j) \cos\sqrt{\lambda_1}\ h - \sqrt{\lambda_1}\ \delta z_1 (t_j) \sin\sqrt{\lambda_1}\ h \end{aligned} \tag{B.4}$$

$\underline{\lambda_1 = 0}$:

$$\begin{aligned} \delta z_1 (t_j + h) &= \delta z_1 (t_j) + \delta \dot{z}_1 (t_j)\ h \\ \delta \dot{z}_1 (t_j + h) &= \delta \dot{z}_1 (t_j) \end{aligned} \tag{B.5}$$

$\underline{\lambda_1 > 0}$:

$$\begin{aligned} \delta z_1 (t_j + h) &= \delta z_1 (t_j) \cosh\sqrt{\lambda_1}\ h + \frac{\delta \dot{z}_1 (t_j)}{\sqrt{\lambda_1}} \sinh\sqrt{\lambda_1}\ h \\ \delta \dot{z}_1 (t_j + h) &= \delta \dot{z}_1 (t_j) \cosh\sqrt{\lambda_1}\ h + \sqrt{\lambda_1}\ \delta z_1 (t_j) \sinh\sqrt{\lambda_1}\ h \end{aligned} \tag{B.6}$$

In these equations the state errors at t_j are the initial conditions, and each of the three component equations has one of the three solution forms above.

Suppose for the moment that $\lambda_1 < 0$ and (B.4) is the analytical relation between the components of the state error at $t_j + h$ and t_j. The transition matrix for these components is obviously

$$C_{t_j+h,\,t_j} = \begin{bmatrix} \cos\sqrt{\lambda_1}\,h & \frac{1}{\sqrt{\lambda_1}}\sin\sqrt{\lambda_1}\,h \\ -\sqrt{\lambda_1}\sin\sqrt{\lambda_1}\,h & \cos\sqrt{\lambda_1}\,h \end{bmatrix} \tag{B.7}$$

The Nystrom numerical integration for the state transition matrix is accurate to fourth order. Therefore, the first term after the fourth order term in a Taylor series expansion for each of the elements in (B.7) is approximately equal to the error in the numerical integration. It is readily apparent that the larger errors must be in the integration of the sine elements in (B.7), since the error is of the fifth order in h while the error in the cosine element integration is of the sixth order. The accuracy requirement is that the ratio of the error term to the lowest order term must be not more than 1×10^{-8}. For each of the two sine elements in (B.7) this turns out to be

$$\frac{(\sqrt{\lambda_1}\,h)^5}{5!\sqrt{\lambda_1}\,h} \leq 1 \times 10^{-8}$$

or

$$h \leq \left[\frac{1.2 \times 10^{-6}}{\lambda_1^{\,2}}\right]^{\frac{1}{4}} = \frac{.033}{\sqrt{\lambda_1}} \tag{B.8}$$

The same analysis can be applied to the case $\lambda_1 > 0$ with the same result (B.8) for h. In the case $\lambda_1 = 0$ the implication is that the numerical integration should give negligible errors for any value of h within the interval in which the initial assumption is valid.

To solve for the eigenvalues and eigenvectors of G at each time point is a complex process and is not done in the program. The square root of the sum of the squares of the eigenvalues is used in place of λ_1 in Equation (B.8). It may be shown that

$$\mathrm{tr}\,(G_j G_j^{\,T}) = \sum_{i=1}^{3} \lambda_i^{\,2} \tag{B.9}$$

$\mathrm{tr}\,(GG^T)$ is easily computed in the program. The fourth root of (B.9) is used in place of λ_1 in (B.8), a scaling constant K in place of the fixed constant, and (3.44) then results.

The differential correction to the time step in hyperbolic orbits is determined in the following manner. Note first that h varies approximately with $r^{3/2}$.

$$h \approx \alpha\, r^{3/2} \tag{B.10}$$

Then

$$dh \approx \frac{3}{2}\, \alpha\, r^{1/2}\, dr$$

$$\approx \frac{3}{2}\, \frac{h}{r}\, dr \tag{B.11}$$

Approximate dr by

$$dr \approx v_r h \tag{B.12}$$

Then the corrected time step is

$$h_c = h + dh = h\left(1 + \frac{3}{2}\, \frac{h}{r}\, v_r\right) \tag{B.13}$$

APPENDIX C

ANALYSIS OF THE ERRORS REFLECTED BY THE CC^{-1} AND RMS MATRICES

This Appendix presents an analysis supporting the discussion in Section 2.2.2.2. The purpose of the analysis is to derive a method of inferring the order of magnitude of the state transistion matrix term errors from the CC^{-1} and RMS matrices, that is, to derive Eq. (2.6) and (2.7).

In the analysis, it is necessary to very carefully differentiate between computational errors which enter the computations at different stages. The elements of the state transition matrix C printed in the program output contain errors due to round-off and truncation in the matrix integration. The order of magnitude of these errors is the quantity of interest. In the subsequent computations of the CC^{-1} and RMS matrices in the program, the term errors in C of course propagate, but in addition there are further roundoff errors made in these calculations. The intent of this analysis is to separate the term errors of C from these latter round-off errors. The analysis proceeds as follows:

Define the term errors in the printed state transition matrix by the following relation:

$$C_{ij} = C_{ij_t} + \epsilon_{ij} \qquad i, j = 1, \cdots, 6 \tag{C.1}$$

where

C_{ij} is the printed (computed) ij^{th} element of C.

C_{ij_t} is the true value of the ij^{th} element of C

ϵ_{ij} is the absolute error in the ij^{th} element

The inverse of the state transition matrix C^{-1} is obtained in the program, as shown in Eqs. (2.1) and (2.2), by simple rearrangement of the elements of C and appropriate sign changes. Define the term errors of C^{-1} by the relation:

$$(C^{-1})_{rs} = (C^{-1})_{rs_t} + (\epsilon^{-1})_{rs}, \quad r, s = 1, \cdots, 6 \tag{C.2}$$

where

$(C^{-1})_{rs}$ is the computed value of the rs^{th} element of C^{-1}

$(C^{-1})_{rs_t}$ is the true value of the rs^{th} element of C^{-1}

$(\epsilon^{-1})_{rs}$ is the absolute error in the rs^{th} element of C^{-1}

It is evident that the magnitude of each $(C^{-1})_{rs}$ is equal to the magnitude of some C_{ij}, since the inverse matrix is obtained by rearrangement of the elements of C and some sign changes.

In the program each element of the CC^{-1} matrix is computed according to the formula

$$(CC^{-1})_{km} = \sum_{j=1}^{6} \left[C_{kj} (C^{-1})_{jm} + \epsilon_{ro_j} \right] , \quad k, m = 1, \cdots, 6 \qquad (C.3)$$

where ϵ_{ro_j} is the roundoff error in the j^{th} term of the computation. Substituting (C. 1) and (C. 2):

$$(CC^{-1})_{km} = \sum_{j=1}^{6} \left\{ \left[C_{kj_t} + \epsilon_{kj} \right] \left[(C^{-1})_{jm_t} + (\epsilon^{-1})_{jm} \right] + \epsilon_{ro_j} \right\} \qquad (C.4)$$

Expanding (C. 4) and retaining only first order error term gives

$$(CC^{-1})_{km} = \sum_{j=1}^{6} C_{kj_t} (C^{-1})_{jm_t} + \sum_{j=1}^{6} \left\{ C_{kj_t} (\epsilon^{-1})_{jm} + (C^{-1})_{jm_t} \epsilon_{kj} + \epsilon_{ro_j} \right\}$$

$$= \delta_{km} + \sum_{j=1}^{6} \left\{ C_{kj_t} (\epsilon^{-1})_{jm} + (C^{-1})_{jm_t} \epsilon_{kj} + \epsilon_{ro_j} \right\}$$

$$= \delta_{km} + \sum_{j=1}^{6} C_{kj_t} (C^{-1})_{jm_t} \left\{ \frac{\epsilon_{kj}}{C_{kj_t}} + \frac{(\epsilon^{-1})_{jm}}{(C^{-1})_{jm_t}} + \frac{\epsilon_{ro_j}}{C_{kj_t} (C^{-1})_{jm_t}} \right\}$$

(C. 5)

where

$$\delta_{km} = \begin{cases} 1, & k = m \\ 0, & k \neq m \end{cases}$$

Each element of the RMS matrix is computed in the program according to:

$$(RMS)_{km} = \frac{(CC^{-1})_{km}}{\sqrt{\sum_{j=1}^{6}\left[C_{kj_t}(C^{-1})_{jm_t}\right]^2}} \qquad (C.6)$$

Now we consider the effects of these errors in certain terms of the CC^{-1} matrix. Eqs. (2.1) through (2.3) express CC^{-1} in terms of the M, N, S, and T submatrices of C:

$$C = \begin{bmatrix} M & N \\ S & T \end{bmatrix} \qquad (2.1)$$

$$C^{-1} = \begin{bmatrix} T^T & -N^T \\ -S^T & M^T \end{bmatrix} \qquad (2.2)$$

$$CC^{-1} = \begin{bmatrix} (MT^T & -NS^T) & (-MN^T & +NM^T) \\ (ST^T & -TS^T) & (-SN^T & +TM^T) \end{bmatrix} \qquad (2.3)$$

C.1 Assessing Magnitude of Matrix Multiplication Round off Errors Only

Consider first the upper right and lower left submatrices of Eq. (2.3). It is not difficult to verify that the diagonal terms of these submatrices should be identically zero regardless of the term errors ϵ_{ij} and $(\epsilon^{-1})_{rs}$ in the elements of C, except for roundoff errors which occur in the computation of these diagonal elements. For example, Eq. (C.5) gives for $(CC^{-1})_{14}$:

$$(CC^{-1})_{14} = \left[C_{11_t}(\epsilon^{-1})_{14} + (C^{-1})_{14_t}\epsilon_{11} + \epsilon_{ro_1}\right] + \left[C_{12_t}(\epsilon^{-1})_{24} + (C^{-1})_{24_t}\epsilon_{12} + \epsilon_{ro_2}\right]$$
$$+ \left[C_{13_t}(\epsilon^{-1})_{34} + (C^{-1})_{34_t}\epsilon_{13} + \epsilon_{ro_3}\right] + \left[C_{14_t}(\epsilon^{-1})_{44} + (C^{-1})_{44_t}\epsilon_{14} + \epsilon_{ro_4}\right]$$
$$+ \left[C_{15_t}(\epsilon^{-1})_{54} + (C^{-1})_{54_t}\epsilon_{15} + \epsilon_{ro_5}\right] + \left[C_{16_t}(\epsilon^{-1})_{64} + (C^{-1})_{64_t}\epsilon_{16} + \epsilon_{ro_6}\right] \qquad (C.7)$$

From Eq. (2. 2) it may be seen that

$$
\begin{aligned}
(C^{-1})_{14} &= -C_{14} \Rightarrow (C^{-1})_{14_t} = -C_{14_t}, \quad (\epsilon^{-1})_{14} = -\epsilon_{14} \\
(C^{-1})_{24} &= -C_{15} \Rightarrow (C^{-1})_{24_t} = -C_{15_t}, \quad (\epsilon^{-1})_{24} = -\epsilon_{15} \\
(C^{-1})_{34} &= -C_{16} \Rightarrow (C^{-1})_{34_t} = -C_{16_t}, \quad (\epsilon^{-1})_{34} = -\epsilon_{16} \\
(C^{-1})_{44} &= C_{11} \Rightarrow (C^{-1})_{44_t} = C_{11_t}, \quad (\epsilon^{-1})_{44} = \epsilon_{11} \\
(C^{-1})_{54} &= C_{12} \Rightarrow (C^{-1})_{54_t} = C_{12_t}, \quad (\epsilon^{-1})_{54} = \epsilon_{12} \\
(C^{-1})_{64} &= C_{13} \Rightarrow (C^{-1})_{64_t} = C_{13_t}, \quad (\epsilon^{-1})_{64} = \epsilon_{13}
\end{aligned}
\tag{C. 8}
$$

Substitution of the conditions (C. 8) into (C. 7) gives

$$
(CC^{-1})_{14} = \sum_{j=1}^{6} \epsilon_{ro_j} \tag{C. 9}
$$

the other terms in (C. 7) canceling out pair by pair. The conclusion is that $(CC^{-1})_{14}$ contains the effects of only the roundoff errors made in the matrix multiplication process, and no effects of the term errors in the C and C^{-1} matrices. We would also expect that the roundoff errors in (C. 9) would cancel out pair by pair also, if the computer were to make equal roundoff errors in the positive and negative members of each pair. That this is not true is an observational fact from the program data. The conclusion is that the complementary arithmetic used by the computer causes unequal errors in the positive and negative members of each pair.

We express (C. 9) in terms of the relative roundoff error in each term:

$$
(CC^{-1})_{14} = \sum_{j=1}^{6} C_{1j_t} (C^{-1})_{j4_t} \left[\frac{\epsilon_{ro_j}}{C_{1j_t} (C^{-1})_{j4_t}} \right] \tag{C. 10}
$$

where $\dfrac{\epsilon_{ro_j}}{C_{ij_t} (C^{-1})_{j4_t}}$ is the relative roundoff error in the j^{th} term. This facilitates a simplifying assumption in the analysis. It is a fact that the relative roundoff error in each term always lies in the interval $(0,\ 1 \times 10^{-8})$. We assume that

the relative roundoff errors are independent stationary random variables with identical statistics over this interval. This assumption can certainly be attacked, since the two roundoff errors from each pair of terms in (C. 9) are functionally related. However, it certainly should be valid for establishing a limit to the order of magnitube of the round off errors.

Let σ^2_{rel} denote the variance of the relative round off errors. We also assume that $(CC^{-1})_{14}$ is an unbiased random variable. That this assumption is in fact good can be verified by examination of a typical Mode 1 trajectory computation. Then, the second order statistics of $(CC^{-1})_{14}$ can be written from (C. 10) as

$$\sigma^2_{(CC^{-1})_{14}} = \sum_{j=1}^{6} \left[C_{1j_t} (C^{-1})_{j4_t} \right]^2 \sigma^2_{rel}$$

$$= \sigma^2_{rel} \sum_{j=1}^{6} \left[C_{1j_t} (C^{-1})_{j4_t} \right]^2 \qquad \text{(C. 11)}$$

From Eq. (C. 6), if $(CC^{-1})_{14}$ is an unbiased random variable, then $(RMS)_{14}$ is likewise. Therefore, from (C. 6) we can write

$$\sigma^2_{(CC^{-1})_{14}} = \sigma^2_{(RMS)_{14}} \sum_{j=1}^{6} \left[C_{1j_t} (C^{-1})_{j4_t} \right] \qquad \text{(C. 12)}$$

Finally, from (C. 11) and (C. 12)

$$\sigma^2_{rel} = \sigma^2_{(RMS)_{14}} \qquad \text{(C. 13)}$$

The above analysis can be repeated for the other five terms of the upper right and lower left (3 x 3) submatrices of CC^{-1}. Eq. (C. 13) results in each case with an appropriate change of subscripts. The conclusion is that the variance of the relative roundoff errors can be estimated directly from the appropriate RMS matrix elements.

C. 2 Assessing Term Error Magnitudes

All elements in the CC^{-1} matrix, other than the six considered in the analysis above, contain both term errors and matrix multiplication roundoff errors. These are shown explicitly in Eq. (C. 5). We rewrite (C. 5) as follows, separating

the relative roundoff errors from the term errors:

$$(CC^{-1})_{km} - \delta_{km} = \sum_{j=1}^{6} C_{k_{jt}} (C^{-1})_{jm_t} \left[\frac{\epsilon_{ro_j}}{C_{k_{jt}} (C^{-1})_{jm_t}} \right]$$

$$+ \sum_{j=1}^{6} C_{kj_t} (C^{-1})_{jm_t} \left[\frac{\epsilon_{kj}}{C_{kj_t}} + \frac{(\epsilon^{-1})_{jm}}{(C^{-1})_{jm_t}} \right] \qquad (C.14)$$

We assumed above that the relative roundoff errors (first ter n the right in (C. 14) were independent, unbiased, and characterized by the variance σ^2_{rel}. Similarly, we assume that the relative term errors (second term on the right in (C. 14) are independent, unbiased, and characterized by a variance σ_t^2. These assumptions are valid only for order of magnitude analysis. By means of these assumptions we can write from (C. 14):

$$\sigma^2_{(CC^{-1})_{km}} = \overline{\left[(CC^{-1})_{km} - \delta_{km}\right]^2} = \overline{(CC^{-1})^2_{km}}$$

$$= \left(\sigma^2_{rel} + 2\sigma_t^2\right) \sum_{j=1}^{6} \left[C_{kj_t} (C^{-1})_{jm_t}\right]^2 \qquad (C.15)$$

Eq. (C. 12) can be generalized for the present case in the form

$$\sigma^2_{(CC^{-1})_{km}} = \sigma^2_{(RMS)_{km}} \sum_{j=1}^{6} \left[C_{k_{jt}} (C^{-1})_{jm_t}\right]^2 \qquad (C.16)$$

and we then conclude that

$$\sigma^2_{rel} + 2\sigma_t^2 = \sigma^2_{(RMS)_{km}} \qquad (C.17)$$

We note that the six main diagonal elements of RMS are in general biased, since the corresponding elements of CC^{-1} are expected to be unity. This must be considered in using (C. 17)

In summary, the RMS matrix allows an estimate of the order of magnitude of the term errors in the following manner. From the diagonal elements of the

upper right and lower left (3 x 3) submatrices of RMS an estimate of the variance σ^2_{rel} of the relative roundoff errors in the CC^{-1} matrix multiplication can be obtained by using Eq. (C.13). Then, from other elements of the RMS matrix an estimate of the variance σ^2_t of the relative term errors can be obtained by using (C.17). An example of the procedure is given in Section 2.2.2.2.

REFERENCES

1. McDonald, W.T., " Special-Purpose Space Trajectory Program for Guidance Studies-User's Manual," M.I.T. Experimental Astronomy Laboratory Research Note, RN-8, July 1965.

2. Stern, R.G., "Interplanetary Midcourse Guidance Analysis," Vols. I and II, M.I.T. Experimental Astronomy Laboratory Report, TE-5, June 1963.

3. Battin, R.H., "Astronautical Guidance," McGraw-Hill Book Company, 1964.

4. Peabody, P.R., Scott, J.F., Orozco, E.G., "JPL Ephemeris Tapes E9510, E9511, and E9512," Technical Memorandum No. 33-167, Jet Propulsion Laboratory, 1964.

5. Henrici, P., "Discrete Variable Methods in Ordinary Differential Equations," John Wiley and Sons, Inc., 1962.